44-理解IO_WAIT：IO性能到底是怎么回事儿？

在专栏一开始的时候，我和你说过，在计算机组成原理这门课里面，很多设计的核心思路，都来源于性能。在前面讲解CPU的时候，相信你已经有了切身的感受了。

大部分程序员开发的都是应用系统。在开发应用系统的时候，我们遇到的性能瓶颈大部分都在I/O上。在第36讲讲解局部性原理的时候，我们一起看了通过把内存当作是缓存，来提升系统的整体性能。在第37讲讲解CPU Cache的时候，我们一起看了CPU Cache和主内存之间性能的巨大差异。

然而，我们知道，并不是所有问题都能靠利用内存或者CPU Cache做一层缓存来解决。特别是在这个“大数据”的时代。我们在硬盘上存储了越来越多的数据，一个MySQL数据库的单表有个几千万条记录，早已经不算是什么罕见现象了。这也就意味着，用内存当缓存，存储空间是不够用的。大部分时间，我们的请求还是要打到硬盘上。那么，这一讲我们就来看看硬盘I/O性能的事儿。

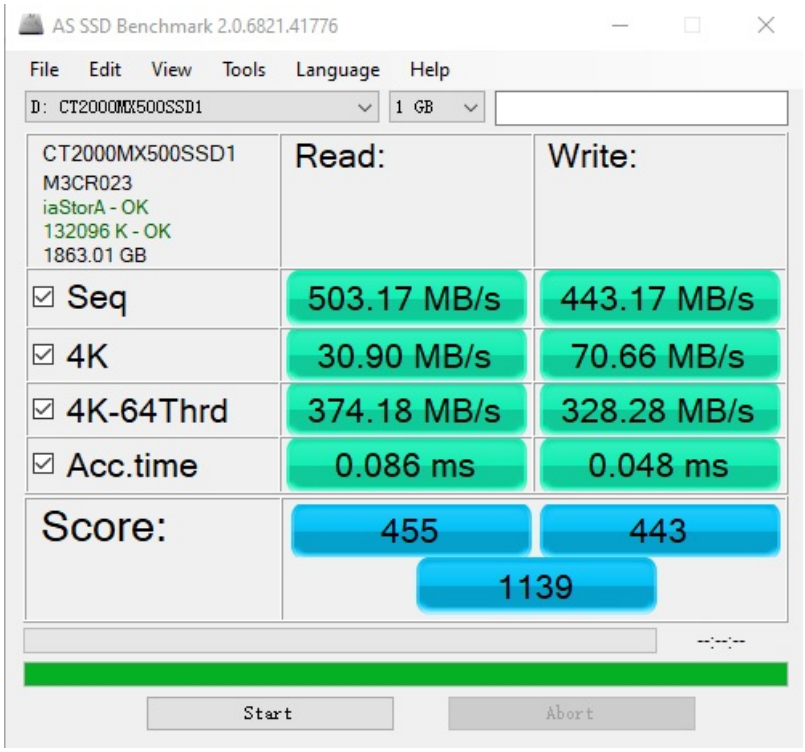
IO性能、顺序访问和随机访问

如果去看硬盘厂商的性能报告，通常你会看到两个指标。一个是**响应时间**（Response Time），另一个叫作**数据传输率**（Data Transfer Rate）。没错，这个和我们在专栏的一开始讲的CPU的性能一样，前面那个就是响应时间，后面那个就是吞吐率了。

我们先来看一看后面这个指标，数据传输率。

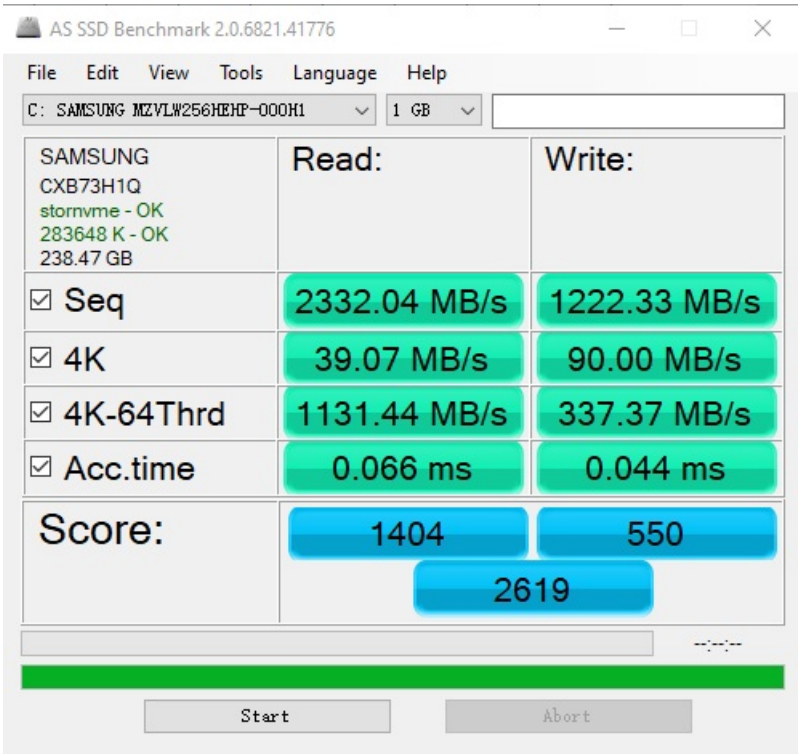
我们现在常用的硬盘有两种。一种是HDD硬盘，也就是我们常说的机械硬盘。另一种是SSD硬盘，一般也被叫作固态硬盘。现在的HDD硬盘，用的是SATA 3.0的接口。而SSD硬盘呢，通常会用两种接口，一部分用的也是SATA 3.0的接口；另一部分呢，用的是PCI Express的接口。

现在我们常用的SATA 3.0的接口，带宽是6Gb/s。这里的“b”是比特。这个带宽相当于每秒可以传输768MB的数据。而我们日常用的HDD硬盘的数据传输率，差不多在200MB/s左右。



这是在我自己的电脑上，运行AS SSD测算SATA接口SSD硬盘性能的结果，第一行的Seq就是顺序读写硬盘

当我们换成SSD的硬盘，性能自然会好上不少。比如，我最近刚把自己电脑的HDD硬盘，换成了一块Crucial MX500的SSD硬盘。它的数据传输速率能到差不多500MB/s，比HDD的硬盘快了一倍不止。不过SATA接口的硬盘，差不多到这个速度，性能也就到顶了。因为SATA接口的速度也就这么快。



不过，实际SSD硬盘能够更快，所以我们可以换用PCI Express的接口。我自己电脑的系统盘就是一块使用了PCI Express的三星SSD硬盘。它的数据传输率，在读取的时候就能做到2GB/s左右，差不多是HDD硬盘的10倍，而在写入的时候也能有1.2GB/s。

除了数据传输率这个吞吐率指标，另一个我们关心的指标响应时间，其实也可以在AS SSD的测试结果里面看到，就是这里面的Acc.Time指标。

这个指标，其实就是程序发起一个硬盘的写入请求，直到这个请求返回的时间。可以看到，在上面的两块SSD硬盘上，大概时间都是在几十微秒这个级别。如果你去测试一块HDD的硬盘，通常会在几毫秒到十几毫秒这个级别。这个性能的差异，就不是10倍了，而是在几十倍，乃至几百倍。

光看响应时间和吞吐率这两个指标，似乎我们的硬盘性能很不错。即使是廉价的HDD硬盘，接收一个来自CPU的请求，也能够在几毫秒时间返回。一秒钟能够传输的数据，也有200MB左右。你想一想，我们平时往数据库里写入一条记录，也就是1KB左右的大小。我们拿200MB去除以1KB，那差不多每秒钟可以插入20万条数据呢。但是这个计算出来的数字，似乎和我们日常的经验不符合啊？这又是为什么呢？

答案就来自于硬盘的读写。在**顺序读写**和**随机读写**的情况下，硬盘的性能是完全不同的。

我们回头看一下上面的AS SSD的性能指标。你会看到，里面有一个“4K”的指标。这个指标是什么意思呢？它其实就是我们的程序，去随机读取磁盘上某一个4KB大小的数据，一秒之内可以读取到多少数据。

你会发现，在这个指标上，我们使用SATA 3.0接口的硬盘和PCI Express接口的硬盘，性能差异变得很小。这是因为，在这个时候，接口本身的速度已经不是我们硬盘访问速度的瓶颈了。更重要的是，你会发现，即

使我们用PCI Express的接口，在随机读写的时候，数据传输率也只能到40MB/s左右，是顺序读写情况下的几十分之一。

我们拿这个40MB/s和一次读取4KB的数据算一下。

$$40MB / 4KB = 10,000$$

也就是说，一秒之内，这块SSD硬盘可以随机读取1万次的4KB的数据。如果是写入的话呢，会更多一些，90MB /4KB 差不多是2万多次。

这个每秒读写的次数，我们称之为IOPS，也就是每秒输入输出操作的次数。事实上，比起响应时间，我们更关注IOPS这个性能指标。IOPS和DTR（Data Transfer Rate，数据传输率）才是输入输出性能的核心指标。

这是因为，我们在实际的应用开发当中，对于数据的访问，更多的是随机读写，而不是顺序读写。我们平时所说的服务器承受的“并发”，其实是在说，会有很多个不同的进程和请求来访问服务器。自然，它们在硬盘上访问的数据，是很难顺序放在一起的。这种情况下，随机读写的IOPS才是服务器性能的核心指标。

好了，回到我们引出IOPS这个问题的HDD硬盘。我现在要问你了，那一块HDD硬盘能够承受的IOPS是多少呢？其实我们应该已经在第36讲说过答案了。

HDD硬盘的IOPS通常也就在100左右，而不是在20万次。在后面讲解机械硬盘的原理和性能优化的时候，我们还会再来一起看一看，这个100是怎么来的，以及我们可以有哪些优化的手段。

如何定位IO_WAIT?

我们看到，即使是用上了PCI Express接口的SSD硬盘，IOPS也就是在2万左右。而我们的CPU的主频通常在2GHz以上，也就是每秒可以做20亿次操作。

即使CPU向硬盘发起一条读写指令，需要很多个时钟周期，一秒钟CPU能够执行的指令数，和我们硬盘能够进行的操作数，也有好几个数量级的差异。这也是为什么，我们在应用开发的时候往往会说“性能瓶颈在I/O上”。因为很多时候，CPU指令发出去之后，不得不去“等”我们的I/O操作完成，才能进行下一步的操作。

那么，在实际遇到服务端程序的性能问题的时候，我们怎么知道这个问题是不是来自于CPU等I/O来完成操作呢？别着急，我们接下来，就通过top和iostat这些命令，一起来看看CPU到底有没有在等待io操作。

```
# top
```

你一定在Linux下用过 top 命令。对于很多刚刚入门Linux的同学，会用top去看服务的负载，也就是load average。不过，在top命令里面，我们一样可以看到CPU是否在等待IO操作完成。

```
top - 06:26:30 up 4 days, 53 min,  1 user,  load average: 0.79, 0.69, 0.65
Tasks: 204 total,   1 running, 203 sleeping,   0 stopped,   0 zombie
```

```
%Cpu(s): 20.0 us,  1.7 sy,  0.0 ni, 77.7 id,  0.0 wa,  0.0 hi,  0.7 si,  0.0 st
KiB Mem:   7679792 total,  6646248 used,  1033544 free,   251688 buffers
KiB Swap:        0 total,        0 used,        0 free.  4115536 cached Mem
```

top命令的输出结果

在top命令的输出结果里面，有一行是以%CPU开头的。这一行里，有一个叫作wa的指标，这个指标就代表着iowait，也就是CPU等待IO完成操作花费的时间占CPU的百分比。下一次，当你自己的服务器遇到性能瓶颈，load很大的时候，你就可以通过top看一看这个指标。

知道了iowait很大，那么我们就要去看一看，实际的I/O操作情况是什么样的。这个时候，你就可以去用iostat这个命令了。我们输入“iostat”，就能够看到实际的硬盘读写情况。

```
$ iostat
```

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          17.02    0.01    2.18    0.04    0.00   80.76
Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               1.81         2.02        30.87     706768   10777408
```

你会看到，这个命令里，不仅有iowait这个CPU等待时间的百分比，还有一些更加具体的指标了，并且它还是按照你机器上安装的多块不同的硬盘划分的。

这里的tps指标，其实就对应着我们上面所说的硬盘的IOPS性能。而kB_read/s和kB_wrtn/s指标，就对应着我们的数据传输率的指标。

知道实际硬盘读写的tps、kB_read/s和kb_wrtn/s的指标，我们基本上可以判断出，机器的性能是不是卡在I/O上了。那么，接下来，我们就是要找出到底是哪一个进程是这些I/O读写的来源了。这个时候，你需要“iotop”这个命令。

```
$ iotop
```

```
Total DISK READ :       0.00 B/s | Total DISK WRITE :      15.75 K/s
Actual DISK READ:       0.00 B/s | Actual DISK WRITE:      35.44 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
  104 be/3 root        0.00 B/s    7.88 K/s  0.00 %  0.18 % [jbd2/sda1-8]
  383 be/4 root        0.00 B/s    3.94 K/s  0.00 %  0.00 % rsyslogd -n [rs:main Q:Reg]
 1514 be/4 www-data    0.00 B/s    3.94 K/s  0.00 %  0.00 % nginx: worker process
```

通过iotop这个命令，你可以看到具体是哪一个进程实际占用了大量I/O，那么你就可以有的放矢，去优化对应的程序了。

上面的这些示例里，不管是wa也好，tps也好，它们都很小。那么，接下来，我就给你用Linux下，用stress命令，来模拟一个高I/O复杂的情况，来看看这个时候的iowait是怎么样的。

我在一台云平台上的单个CPU核心的机器上输入“stress -i 2”，让stress这个程序模拟两个进程不停地从内存里往硬盘上写数据。

```
$ stress -i 2
```

```
$ top
```

你会看到，在top的输出里面，CPU就有大量的sy和wa，也就是系统调用和iowait。

```
top - 06:56:02 up 3 days, 19:34,  2 users,  load average: 5.99, 1.82, 0.63
Tasks:  88 total,   3 running,  85 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.0 us, 29.9 sy,  0.0 ni,  0.0 id, 67.2 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1741304 total,  1004404 free,   307152 used,   429748 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  1245700 avail Mem
```

```
$ iostat 2 5
```

如果我们通过iostat，查看硬盘的I/O，你会看到，里面的tps很快就到了4万左右，占满了对应硬盘的IOPS。

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           5.03    0.00   67.92   27.04    0.00    0.00
Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda           39762.26         0.00         0.00          0          0
```

如果这个时候我们去看一看iotop，你就会发现，我们的I/O占用，都来自于stress产生的两个进程了。

```
$ iotop
```

```
Total DISK READ :       0.00 B/s | Total DISK WRITE :       0.00 B/s
Actual DISK READ:       0.00 B/s | Actual DISK WRITE:       0.00 B/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
29161 be/4 xuwenhao    0.00 B/s    0.00 B/s  0.00 % 56.71 % stress -i 2
29162 be/4 xuwenhao    0.00 B/s    0.00 B/s  0.00 % 46.89 % stress -i 2
    1 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % init
```

相信到了这里，你也应该学会了怎么通过top、iostat以及iotop，一步一步快速定位服务器端的I/O带来的性能瓶颈了。你也可以自己通过Linux的man命令，看一看这些命令还有哪些参数，以及通过stress来模拟其他更多不同的性能压力，看看我们的机器负载会发生什么变化。

总结延伸

这一讲里，我们从硬盘的两个核心指标，响应时间和数据传输率，来理解和研究I/O的性能问题。你也自己可以通过as ssd这样的性能评测软件，看一看自己的硬盘性能。

在顺序读取的情况下，无论是HDD硬盘还是SSD硬盘，性能看起来都是很不错的。不过，等到进行随机读取测试的时候，硬盘的性能才能见了真章。因为在大部分的应用开发场景下，我们关心的并不是在顺序读写下的数据量，而是每秒钟能够进行输入输出的操作次数，也就是IOPS这个核心性能指标。

你会发现，即使是使用PCI Express接口的SSD硬盘，IOPS也就只是到了2万左右。这个性能，和我们CPU的每秒20亿次操作的能力比起来，可就差得远了。所以很多时候，我们的程序对外响应慢，其实都是CPU在等待I/O操作完成。

在Linux下，我们可以通过top这样的命令，来看整个服务器的整体负载。在应用响应慢的时候，我们可以先通过这个指令，来看CPU是否在等待I/O完成自己的操作。进一步地，我们可以通过iostat这个命令，来看到各个硬盘这个时候的读写情况。而 iotop 这个命令，能够帮助我们定位到到底是哪一个进程在进行大量的I/O操作。

这些命令的组合，可以快速帮你定位到是不是我们的程序遇到了I/O的瓶颈，以及这些瓶颈来自于哪些程序，你就可以根据定位的结果来优化你自己的程序了。

推荐阅读

关于IO_WAIT的文章，在互联网上已经有不少了。你可以读一读这一篇Understanding IOPS Latency and Storage Performance，进一步理解一下什么是IOPS和IO_WAIT。

课后思考

你能去下载一个AS SSD软件，测试一下你自己硬盘的性能吗？特别是如果你手上还有HDD硬盘的话，可以尝试测试一下HDD硬盘的性能是怎么样的。

在上面的性能指标上，我们已经讲解了Seq，4K以及Acc.Time这三个指标，那么4K-Thrd这个指标又是什么意思呢？测试这个指标对应的应用场景又是怎么样的呢？

请你研究一下，把你得到的答案写在留言区，和大家一起分享讨论吧。另外，如果有收获，也欢迎你把这篇

文章分享给你的朋友。

精选留言：

- 有铭 2019-08-07 10:55:09
  才注意到，硬盘的随机读的性能是不如随机写的。我以前一直以为是反过来的，但是为什么呢？按这个现象，原来硬盘类设备是“写多读少”的设计思路？ [1赞]

- humor 2019-08-07 15:55:44
  我实验的结果也是sy高，但是wa是0.0。不知道为什么

- 许童童 2019-08-07 15:10:57
  老师你好，我有一台云主机，怎么样才能测出硬盘的性能，比如最大的IOPS?

- 许童童 2019-08-07 15:00:37
  那么 4K-Thrd 这个指标又是什么意思呢？测试这个指标对应的应用场景又是怎么样的呢?
  这个指标是同时读写64个文件时的吞吐量，应用场景主要是服务器端需要处理大量并发请求。

- bro. 2019-08-07 14:54:46
  4K-64Thrd是随机64队列深度测试，软件则会生成64个16MB大小的测试文件（共计1GB），然后同时以4KB的单位尺寸，同时在这64个文件中进行写入和读取测试，最后依然以平均成绩为结果!

- 许童童 2019-08-07 14:54:27
  老师你好，我在CENTOS下用Stress命令，IO_WAIT并没有上去，是什么原因?

- haer 2019-08-07 11:42:51
  关于 stress -i 2 命令
  在VMware虚拟机中测试，99 sy，0.0 wa
  在vps中测试，24.3 sy，73.1 wa
  为什么会这样呢?

- we 2019-08-07 09:56:25
  我在本地虚拟机里面测试，iowait 没有升高。而是平均负载与sy 特别高。

- we 2019-08-07 09:27:38
  老师，stress 命令在centos 下有三个类似的包。是哪个呢？ stress-ng,http ress,

- we 2019-08-07 09:08:53
  pci-e 的ssd 磁盘iops 才20000。那云平台的云磁盘是不是更低了。