iostat来对linux硬盘IO性能进行了解Linux认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_iostat_E6_9D_A5_E5_c103_644686.htm 以前一直不太会用这个参数。现在认真研究了一下iostat，因为刚好有台重要的服务器压力高,所以放上来分析一下.下面这台就是IO有压力过大的服务器 # iostat -x 1 10 Linux 2.6.18-92.el5xen 02/03/2009 avg-cpu: %user %nice %system %iowait %steal %idle 1.10 0.00 4.82 39.54 0.07 54.46 Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s avgrq-sz avgqu-sz await svctm %util sda 0.00 3.50 0.40 2.50 5.60 48.00 18.48 0.00 0.97 0.97 0.28 sdb 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 sdc 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 sdd 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 sde 0.00 0.10 0.30 0.20 2.40 2.40 9.60 0.00 1.60 1.60 0.08 sdf 17.40 0.50 102.00 0.20 12095.20 5.60 118.40 0.70 6.81 2.09 21.36 sdg 232.40 1.90 379.70 0.50 76451.20 19.20 201.13 4.94 13.78 2.45 93.16 rrqm/s: 每秒进行 merge 的读操作数目。即 delta(rmerge)/s wrqm/s: 每秒进行 merge 的写操作数目。即 delta(wmerge)/s r/s: 每秒完成的读 I/O 设备次数。即 delta(rio)/s w/s: 每秒完成的写 I/O 设备次数。即 delta(wio)/s rsec/s: 每秒读扇区数。即 delta(rsect)/s wsec/s: 每秒写扇区数。即 delta(wsect)/s rkB/s: 每秒读K字节数。是 rsect/s 的一半，因为每扇区大小为512字节。(需要计算) wkB/s: 每秒写K字节数。是 wsect/s 的一半。(需要计算) avgrq-sz: 平均每次设备I/O操作的数据大小 (扇区)。delta(rsect wsect)/delta(rio wio) avgqu-sz: 平均I/O队列长度。即 delta(aveq)/s/1000 (因为aveq的单位为毫秒)。 await: 平均每次设备I/O操作的等待时间 (毫秒)。即

delta(ruse wuse)/delta(rio wio) svctm: 平均每次设备I/O操作的服务时间 (毫秒)。即 delta(use)/delta(rio wio) %util: 一秒中有百分之多少的时间用于 I/O 操作，或者说一秒中有多少时间 I/O 队列是非空的。即 delta(use)/s/1000 (因为use的单位为毫秒) 如果 %util 接近 100%，说明产生的I/O请求太多，I/O系统已经满负荷，该磁盘 可能存在瓶颈。 idle小于70% IO压力就较大了,一般读取速度有较多的wait. 同时可以结合vmstat 查看查看b参数(等待资源的进程数)和wa参数(IO等待所占用的CPU时间的百分比,高过30%时IO压力高) 另外还可以参考 svctm 一般要小于 await (因为同时等待的请求的等待时间被重复计算了)，svctm 的大小一般和磁盘性能有关，CPU/内存的负荷也会对其有影响，请求过多也会间接导致 svctm 的增加。await 的大小一般取决于服务时间(svctm) 以及 I/O 队列的长度和 I/O 请求的发出模式。如果 svctm 比较接近 await，说明 I/O 几乎没有等待时间；如果 await 远大于 svctm，说明 I/O 队列太长，应用得到的响应时间变慢，如果响应时间超过了用户可以容许的范围，这时可以考虑更换更快的磁盘，调整内核 elevator 算法，优化应用，或者升级 CPU。 队列长度(avgqu-sz)也可作为衡量系统 I/O 负荷的指标，但由于 avgqu-sz 是按照单位时间的平均值，所以不能反映瞬间的 I/O 洪水。 别人一个不错的例子.(I/O 系统 vs. 超市排队) 举一个例子，我们在超市排队 checkout 时，怎么决定该去哪个交款台呢? 首当是看排的队人数，5个人总比20人要快吧? 除了数人头，我们也常常看看前面人购买的东西多少，如果前面有个采购了一星期食品的大妈，那么可以考虑换个队排了。还有就是收银员的速度了，如果碰上了连钱都点不清楚的新手，那

就有的等了。另外，时机也很重要，可能 5 分钟前还人满为患的收款台，现在已是人去楼空，这时候交款可是很爽啊，当然，前提是那过去的 5 分钟里所做的事情比排队要有意义 (不过我还没发现什么事情比排队还无聊的)。 I/O 系统也和超市排队有很多类似之处: r/s w/s 类似于交款人的总数 平均队列长度(avgqu-sz)类似于单位时间里平均排队人的个数 平均服务时间(svctm)类似于收银员的收款速度 平均等待时间(await)类似于平均每人的等待时间 平均I/O数据(avgrq-sz)类似于平均每人所买的东西多少 I/O 操作率 (%util)类似于收款台前有人排队的时间比例。 我们可以根据这些数据分析出 I/O 请求的模式，以及 I/O 的速度和响应时间。 下面是别人写的这个参数输出的分析 # iostat -x 1 avg-cpu: %user %nice %sys %idle 16.24 0.00 4.31 79.44 Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util /dev/cciss/c0d0 0.00 44.90 1.02 27.55 8.16 579.59 4.08 289.80 20.57 22.35 78.21 5.00 14.29 /dev/cciss/c0d0p1 0.00 44.90 1.02 27.55 8.16 579.59 4.08 289.80 20.57 22.35 78.21 5.00 14.29 /dev/cciss/c0d0p2 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 上面的 iostat 输出表明秒有 28.57 次设备 I/O 操作: 总IO(io)/s = r/s(读) w/s(写) = 1.02 27.55 = 28.57 (次/秒) 其中写操作占了主体 (w:r = 27:1)。 平均每次设备 I/O 操作只需要 5ms 就可以完成，但每个 I/O 请求却需要等上 78ms，为什么? 因为发出的 I/O 请求太多 (每秒钟约 29 个)，假设这些请求是同时发出的，那么平均等待时间可以这样计算: 平均等待时间 = 单个 I/O 服务时间 * (1 2 … 请求总数-1) / 请求总数 应用到上面的例子: 平均等待时间 = 5ms * (1 2 … 28)/29 = 70ms，和 iostat 给出的78ms 的平均等待时间

很接近。这反过来表明 I/O 是同时发起的。 每秒发出的 I/O 请求很多 (约 29 个)，平均队列却不长 (只有 2 个 左右)，这表明这 29 个请求的到来并不均匀，大部分时间 I/O 是空闲的。一秒中有 14.29% 的时间 I/O 队列中是有请求的，也就是说，85.71% 的时间里 I/O 系统无事可做，所有 29 个 I/O 请求都在142毫秒之内处理掉了。 delta(ruse wuse)/delta(io) = await = 78.21 =&gt. delta(ruse wuse)/s =78.21 * delta(io)/s = 78.21*28.57 = 2232.8，表明每秒内的I/O请求总共需要等待2232.8ms。所以平均队列长度应为 2232.8ms/1000ms = 2.23，而 iostat 给出的平均队列长度 (avgqu-sz) 却为 22.35，为什么?! 因为 iostat 中有 bug，avgqu-sz 值应为 2.23，而不是 22.35。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com