I/O statistics fields

Since 2.4.20 (and some versions before, with patches), and 2.5.45, more extensive disk statistics have been introduced to help measure disk activity. Tools such as `sar` and `iostat` typically interpret these and do the work for you, but in case you are interested in creating your own tools, the fields are explained here.

In 2.4 now, the information is found as additional fields in `/proc/partitions`. In 2.6 and upper, the same information is found in two places: one is in the file `/proc/diskstats`, and the other is within the sysfs file system, which must be mounted in order to obtain the information. Throughout this document we'll assume that sysfs is mounted on `/sys`, although of course it may be mounted anywhere. Both `/proc/diskstats` and sysfs use the same source for the information and so should not differ.

Here are examples of these different formats:

```
2.4:
   3     0   39082680 hda 446216 784926 9550688 4382310 424847 312726 5922052 19310380 0 3376340 2370516
   3     1    9221278 hda1 35486 0 35496 38030 0 0 0 0 0 38030 38030

2.6+ sysfs:
   446216 784926 9550688 4382310 424847 312726 5922052 19310380 0 3376340 23705160
   35486    38030    38030    38030

2.6+ diskstats:
   3    0   hda 446216 784926 9550688 4382310 424847 312726 5922052 19310380 0 3376340 23705160
   3    1   hda1 35486 38030 38030 38030

4.18+ diskstats:
   3    0   hda 446216 784926 9550688 4382310 424847 312726 5922052 19310380 0 3376340 23705160 0 0 0 0
```

On 2.4 you might execute `grep 'hda ' /proc/partitions`. On 2.6+, you have a choice of `cat /sys/block/hda/stat` or `grep 'hda ' /proc/diskstats`.

The advantage of one over the other is that the sysfs choice works well if you are watching a known, small set of disks. `/proc/diskstats` may be a better choice if you are watching a large number of disks because you'll avoid the overhead of 50, 100, or 500 or more opens/closes with each snapshot of your disk statistics.

In 2.4, the statistics fields are those after the device name. In the above example, the first field of statistics would be 446216. By contrast, in 2.6+ if you look at `/sys/block/hda/stat`, you'll find just the 15 fields, beginning with 446216. If you look at `/proc/diskstats`, the 15 fields will be preceded by the major and minor device numbers, and device name. Each of these formats provides 15 fields of statistics, each meaning exactly the same things. All fields except field 9 are cumulative since boot. Field 9 should go to zero as I/Os complete; all others only increase (unless they overflow and wrap). Wrapping might eventually occur on a very busy or long-lived system; so applications should be prepared to deal with it. Regarding wrapping, the types of the fields are either unsigned int (32 bit) or unsigned long (32-bit or 64-bit, depending on your machine) as noted per-field below. Unless your observations are very spread in time, these fields should not wrap twice before you notice it.

Each set of stats only applies to the indicated device; if you want system-wide stats you'll have to find all the devices and sum them all up.

Field 1 -- # of reads completed (unsigned long)

> This is the total number of reads completed successfully.

Field 2 -- # of reads merged, field 6 -- # of writes merged (unsigned long)

> Reads and writes which are adjacent to each other may be merged for efficiency. Thus two 4K reads may become one 8K read before it is ultimately handed to the disk, and so it will be counted (and queued) as only one I/O. This field lets you know how often this was done.

Field 3 -- # of sectors read (unsigned long)

> This is the total number of sectors read successfully.

Field 4 -- # of milliseconds spent reading (unsigned int)

> This is the total number of milliseconds spent by all reads (as measured from blk_mq_alloc_request() to __blk_mq_end_request()).

Field 5 -- # of writes completed (unsigned long)

> This is the total number of writes completed successfully.

Field 6 -- # of writes merged (unsigned long)

> See the description of field 2.

Field 7 -- # of sectors written (unsigned long)

> This is the total number of sectors written successfully.

Field 8 -- # of milliseconds spent writing (unsigned int)

This is the total number of milliseconds spent by all writes (as measured from blk_mq_alloc_request() to __blk_mq_end_request()).

Field 9 -- # of I/Os currently in progress (unsigned int)

The only field that should go to zero. Incremented as requests are given to appropriate struct request_queue and decremented as they finish.

Field 10 -- # of milliseconds spent doing I/Os (unsigned int)

This field increases so long as field 9 is nonzero.

Since 5.0 this field counts jiffies when at least one request was started or completed. If request runs more than 2 jiffies then some I/O time might be not accounted in case of concurrent requests.

Field 11 -- weighted # of milliseconds spent doing I/Os (unsigned int)

This field is incremented at each I/O start, I/O completion, I/O merge, or read of these stats by the number of I/Os in progress (field 9) times the number of milliseconds spent doing I/O since the last update of this field. This can provide an easy measure of both I/O completion time and the backlog that may be accumulating.

Field 12 -- # of discards completed (unsigned long)

This is the total number of discards completed successfully.

Field 13 -- # of discards merged (unsigned long)

See the description of field 2

Field 14 -- # of sectors discarded (unsigned long)

This is the total number of sectors discarded successfully.

Field 15 -- # of milliseconds spent discarding (unsigned int)

This is the total number of milliseconds spent by all discards (as measured from blk_mq_alloc_request() to __blk_mq_end_request()).

Field 16 -- # of flush requests completed

This is the total number of flush requests completed successfully.

Block layer combines flush requests and executes at most one at a time. This counts flush requests executed by disk. Not tracked for partitions.

Field 17 -- # of milliseconds spent flushing

This is the total number of milliseconds spent by all flush requests.

To avoid introducing performance bottlenecks, no locks are held while modifying these counters. This implies that minor inaccuracies may be introduced when changes collide, so (for instance) adding up all the read I/Os issued per partition should equal those made to the disks ... but due to the lack of locking it may only be very close.

In 2.6+, there are counters for each CPU, which make the lack of locking almost a non-issue. When the statistics are read, the per-CPU counters are summed (possibly overflowing the unsigned long variable they are summed to) and the result given to the user. There is no convenient user interface for accessing the per-CPU counters themselves.

Since 4.19 request times are measured with nanoseconds precision and truncated to milliseconds before showing in this interface.

Disks vs Partitions

There were significant changes between 2.4 and 2.6+ in the I/O subsystem. As a result, some statistic information disappeared. The translation from a disk address relative to a partition to the disk address relative to the host disk happens much earlier. All merges and timings now happen at the disk level rather than at both the disk and partition level as in 2.4. Consequently, you'll see a different statistics output on 2.6+ for partitions from that for disks. There are only *four* fields available for partitions on 2.6+ machines. This is reflected in the examples above.

Field 1 -- # of reads issued
This is the total number of reads issued to this partition.
Field 2 -- # of sectors read
This is the total number of sectors requested to be read from this partition.
Field 3 -- # of writes issued
This is the total number of writes issued to this partition.
Field 4 -- # of sectors written
This is the total number of sectors requested to be written to this partition.

Note that since the address is translated to a disk-relative one, and no record of the partition-relative address is kept, the subsequent success or failure of the read cannot be attributed to the partition. In other words, the number of reads for partitions is counted slightly before time of queuing for partitions, and at completion for whole disks. This is a subtle distinction that is probably uninteresting for most cases.

More significant is the error induced by counting the numbers of reads/writes before merges for partitions and after for disks. Since a

typical workload usually contains a lot of successive and adjacent requests, the number of reads/writes issued can be several times higher than the number of reads/writes completed.

In 2.6.25, the full statistic set is again available for partitions and disk and partition statistics are consistent again. Since we still don't keep record of the partition-relative address, an operation is attributed to the partition which contains the first sector of the request after the eventual merges. As requests can be merged across partition, this could lead to some (probably insignificant) inaccuracy.

Additional notes

In 2.6+, sysfs is not mounted by default. If your distribution of Linux hasn't added it already, here's the line you'll want to add to your `/etc/fstab`:

```
none /sys sysfs defaults 0 0
```

In 2.6+, all disk statistics were removed from `/proc/stat`. In 2.4, they appear in both `/proc/partitions` and `/proc/stat`, although the ones in `/proc/stat` take a very different format from those in `/proc/partitions` (see proc(5), if your system has it.)

-- ricklind@us.ibm.com