

# Eric Bergen

You will probably want some waders, a pick axe, and one of those hats with a light on it before you go in here.

---

## The difference between %iowait from sar and %util from iostat

2/25/2008, 11:09 am

Recently I needed to explain the difference between %iowait from sar and %util from iostat. They both measure outstanding i/o requests but their method for measurement is slightly different. From the sar man page iowait is defined as:

**iowait:**  
Percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.

iostat has a slightly different definition of it's disk metric which is %util (percent utilization).

**%util:**  
Percentage of CPU time during which I/O requests were issued to the device (bandwidth utilization for the device). Device saturation occurs when this value is close to 100%.

The definitions are very similar but there is a slight difference. %iowait uses the per cpu timings from /proc/stat. /proc/stat breaks down cpu time into four parts. The fields are defined as:

- user: normal processes executing in user mode
- nice: niced processes executing in user mode
- system: processes executing in kernel mode
- idle: twiddling thumbs
- iowait: waiting for I/O to complete
- irq: servicing interrupts
- softirq: servicing softirqs
- steal: involuntary wait

The only one important for this discussion is iowait. Inside the kernel iowait time is basically time processes spend waiting for requests that have been sent into the i/o scheduler.

iostat's %util is measured using /proc/diskstats which keeps track of the number of milliseconds devices spent with outstanding i/o requests. iostat compares uptime of the server to the number of milliseconds spent doing i/o to determine how busy the disk is.

The main difference I can see between these two approaches is that sar will count the time

a i/o request spent in the i/o scheduler where iostat won't.

A few test runs show wildly different results from the two different tools. While running bonnie++, iostat, and sar in parallel some times iostat returns 100 %util and sar returns 25% iowait. During the read phase of bonnie++ iostat can return 100% where sar returns 1%. It's very strange...

I'm still a bit unclear about all the differences. I'll update this post when I learn more.

Category: Geek | Comment (RSS) | Trackback

### 3 Comments

1. *Sean* says:

2/25/2008 at 11:44 am



iowait is effectively idle time (in 2.4 kernels, it was rolled up into idle).

I'd be interested in seeing the other numbers (sys, user, idle) for the bonnie++ tests... If the disks are maxed and only 1% iowait, then user% is probably running at ~99%.

Sean

2. *Joerg* says:

3/5/2008 at 12:41 pm



IMO, "%iowait" is seen from a CPU point of view: No task was runnable, but at least one was waiting for disk IO (which may include paging, AFAIK). If the disk(s) were infinitely fast, the transfer would be done, and the task could continue. Slightly different: "%iowait" is the (theoretical) limit to throughput improvement you could achieve by getting rid of disk IO (all files in memory, sufficient RAM not to page, ...).


"%util" shows you whether the disk(s) can keep up with the requests issued. When it approaches 100 % for one disk, you \*have\* to distribute that disk's load to multiple spindles.

You can have heavy disk utilization without waiting for them (low "%iowait") if you have some CPU-consuming background task which does not wait for disk IO. It would keep your CPU(s) busy while other tasks wait for the disk. Computing PI would be a candidate, I assume.

Joerg

3. *high load average, no cpu or ram usage - cPanel Forums* says:

8/21/2009 at 5:19 am

[...] links also helped me get to grips with iostat: iostat and disk utilization monitoring nirvana Eric Bergen  The difference between %iowait from sar and %util from iostat

Let me know how you get on. [...]