# Demystifying I/O Wait

**ABSTRACT - A great deal of controversy exists around the interpretation of the I/O wait metric in AIX. This number shows up at the rightmost "wa" column in vmstat output, the "% iowait" column in iostat, the %wio column in the sar -P , and the ascii bar graph titled "wait" in topas. Confusion exists when I/O wait is evaluated for performance or capacity planning as to whether this number should be considered CPU cycles that are used or cycles that should be added to the system idle time indicating unused capacity. This paper will explain how this metric is captured and calculated as well as provide a "case study" example to illustrate the effects.**

**A review of some of the basic AIX functions will assist in a better understanding of how the I/O wait value is collected and calculated. The AIX scheduler, the CPU "queues", the CPU states, and the idle or wait process, will be discussed.**

**The scheduler is a part of the AIX kernel that is tasked with making sure the individual CPUs have work to do and in the case where there are more runnable jobs (threads) than CPUs, to make sure each one gets its fair share of the CPU resource. The system contains a hardware timer which generates 100 interrupts/second. This interrupt will then dispatch the kernel scheduler process which runs at a fixed priority of 16. The scheduler will first charge the running thread with the 10 millisecond time slice and then dispatch another thread (context switch) of equal or higher priority on that CPU assuming there are other runnable threads. This short term CPU usage**

**Partial ps command output showing short term CPU usage in "C' column**

```
#ps -aekl
       F S      UID   PID  PPID   C PRI NI ADDR    SZ    WCHAN    TTY  TIME CMD
     303 A        0     0     0 120  16 -- 12012    12               -  4:17 swapper

  200003 A        0     1     0   0  60 20 c00c    732               -  0:22 init
     303 A        0   516     0 120 127 -- 13013     8               - 31972:23 kproc

     303 A        0   774     0 120 127 -- 14014     8               - 31322:34 kproc

     303 A        0  1032     0   0  16 -- 17017    12               -  0:00 kproc

     303 A        0  1290     0   0  36 -- 1e01e    16               -  0:32 kproc

     303 A        0  1548     0   0  37 -- 1f01f    64        *      -  5:09 kproc

     303 A        0  1806     0   0  60 -- c02c     16 3127c558      -  0:04 kproc

  240001 A        0  2638     1   0  60 20 12212   108 3127ab98      - 102:04 syncd
```

**is reported in the "C" column when including a -l option with the ps command.**

```
#
```

**One hundred times a second, the scheduler will take the process that is currently running on each CPU, and increment the "C" value by one. It will then recalculate that processes priority and rescan the process table looking for the next process to dispatch. If there are no runnable processes the scheduler will dispatch the "idle" kernel process. There are one of these assigned to each CPU and are bound to that particular processor. The following output shows a four way system with four wait processes each bound to a CPU.**

```
THREAD TABLE :

SLT ST    TID      PID    CPUID  POLICY PRI CPU    EVENT    PROCNAME
  0 s       3        0  unbound    FIFO  10  78            swapper
        flags:  kthread
  1 s     103        1  unbound   other  3c   0              init
        flags:  local wakeonsig cdefer
 unknown: 0x10000
  2 r     205      204        0    FIFO  ff  78              wait
        flags:  funnel kthread
  3 r     307      306        1    FIFO  ff  78              wait
        flags:  funnel kthread
  4 r     409      408        2    FIFO  ff  78              wait
        flags:  funnel kthread
  5 r     50b      50a        3    FIFO  ff  78              wait
        flags:  funnel kthread
  6 s     60d      60c  unbound      RR  11  2b            reaper
```

**Also notice that the wait process priority is 0Xff. The MSB has been turned off to give a priority range of 0-127 on AIX 5.1 and lower. In AIX 5.2 and higher the range of priorities has been increased to 255 to allow more granularity for control when using Workload Manager (WLM).**

**If there are no processes to dispatch, the scheduler will dispatch the “wait” process which will run until any other process becomes runnable at which time it will immediately be dispatched since it will always have a higher priority. The wait processes only job is to increment the counters that report if that particular processor is “idle” or “waiting for I/O”. It is important to remember that the “waiting for I/O” metric is incremented by the idle process. The decision on whether the idle process decides to increment the “idle” counter or the “waiting for I/O counter depends on whether there is a process sitting in the blocked queue. Processes which are runnable but waiting on data from a disk are placed on the blocked queue to wait for their data. If no processes are sitting on that particular processors blocked queue, then the wait process will charge the time to “idle”. If there are one or more processes on that particular processors blocked queue, then the system**

**charges the time to “waiting for I/O”. Waiting for I/O is considered to be a special case of idle and therefore the percentage of time spent in waiting for I/O is usable for process to perform work.**

**A case study will be presented to illustrate this concept. Consider a single CPU system the has two tasks to perform. Task a is a CPU intensive program and task B is an I/O intensive program. The effects of these programs on the vmstat output will be considered separately and then combined.**

**Task “A” which is CPU intensive is run on a single CPU system, the majority of the CPU time will be spent in the “user” (us) mode. The vmstat output below reflects the effects of a single process running.**

```
$ vmstat 1
kthr     memory             page                     faults         cpu
-----  ----------- ------------------------    ------------  -----------
 r  b   avm   fre     re  pi  po  fr   sr  cy  in   sy  cs   us sy id wa
 1  0 106067 164605   0   0   0   0   23   0   232  835 411  99  0  0  0
 1  0 106072 164600   0   0   0   0    0   0   239 2543 413  99  1  0  0
 1  0 106072 164600   0   0   0   0    0   0   234 2425 403  99  0  0  0
 1  0 106072 164600   0   0   0   0    0   0   235 2426 405  98  2  0  0
 1  0 106072 164600   0   0   0   0    0   0   241 2572 428  99  1  0  0
 1  0 106072 164600   0   0   0   0    0   0   233 2490 475  99  0  0  0
```

**Task “B” which is I/O intensive is run on a single CPU system, the majority of the CPU time will be spent in the “waiting for I/O” (wa) mode. The vmstat output below reflects the effects of a single process running.**

```
$ vmstat 1
kthr     memory             page                     faults         cpu
-----  ----------- ------------------------    ------------  -----------
 r  b   avm   fre     re  pi  po  fr   sr  cy  in   sy  cs   us sy id wa
 0  1 106067 164605   0   0   0   0   23   0   232  835 411   0  1  0  99
 0  1 106072 164600   0   0   0   0    0   0   239 2543 413   0  1  0  99
 0  1 106072 164600   0   0   0   0    0   0   234 2425 403   0  1  0  99
 1  1 106072 164600   0   0   0   0    0   0   235 2426 405   0  2  0  98
 0  1 106072 164600   0   0   0   0    0   0   241 2572 428   0  1  0  99
 0  1 106072 164600   0   0   0   0    0   0   233 2490 475   0  1  0  99
```

**If while Task "B" which is I/O intensive is running, task "A" is started on a single CPU system, the majority of the CPU time will be spent in the "user" (us) mode. This shows that all of the CPU cycles spent in the "waiting for I/O" mode have been recovered and are usable by other processes. The vmstat output below reflects the effects of running a CPU intensive program and an I/O intensive simultaneously.**

```
$ vmstat 1
kthr     memory             page                     faults        cpu
-----  ----------- ------------------------    ------------  -----------
 r  b   avm   fre     re  pi  po  fr   sr  cy  in   sy  cs    us sy id wa
 1  1 106067 164605   0   0   0   0   23   0   232  835 411   99  0  0  0
 1  1 106072 164600   0   0   0   0    0   0   239 2543 413   99  2  0  0
 1  1 106072 164600   0   0   0   0    0   0   234 2425 403   99  0  0  0
 2  1 106072 164600   0   0   0   0    0   0   235 2426 405   98  1  0  0
 1  1 106072 164600   0   0   0   0    0   0   241 2572 428   99  1  0  0
 1  1 106072 164600   0   0   0   0    0   0   233 2490 475   99  0  0  0
```

**One item to note from this example. I/O bound systems cannot always be determined by looking at the "waiting for I/O" metrics only. A busy system can mask the effects of I/O bottlenecks. To determine if an I/O bottleneck exists, the blocked queue as well as the output from iostat must also be considered.**