White Paper: Task Manager and Under-reporting

Task Manager Under-reporting

Your processors are busier than you think...

A service of TMurgent Technologies. Link to: TMurgent Technologies Home Page

The Task Manager provides a reasonable estimate of CPU activity for most uses. It does, however, have a consistent bias that under-reports actual usage. The under-reporting can be as much as 5% on some systems. We at TMurgent Technologies have spent a considerable effort to investigate these differences, and present our findings here.

In this paper, we will provide a brief overview of CPU utilization measurement in general, followed by an explanation of two ways in which the Task Manager under-reports this measurement.

CPU Utilization Measurement

In a multi-tasking operating system one often wants to understand the impact on resources that various software components - either applications or OS components - use. This is especially true of the CPU resources.

While it is possible to provide a complete and accurate accounting of CPU utilization, such an accounting either requires additional watchdog hardware separate from the CPU(s), or a considerable overhead. Instead, OS design has evolved to use a method of estimation. While not perfect, estimation provides a very good measurement, especially when viewed over time.

Typically within the kernel of the OS, an accounting is made each time the lowest level clock tick occurs. When the tick occurs, a check is made to determine what thread/process was active in the processor at that time. The CPU usage counter for the thread/process is incremented to represent the amount of time between ticks.

While it is possible (perhaps even likely) that this thread/process was not executing for the entire period between ticks, this is a reasonable approximation that provides fairly accurate results in the long run.

When the clock tick occurs, it may happen that the processor was processing an interrupt. It is likely that the interrupt will have nothing to do with the active process, so the question occurs as to how to account for the time. There are basically three possibilities:

- Determine what thread/process is responsible for the interrupt and assign the period to it.
- Account for interrupts separately from thread/process usage.
- Assign the period to the task that was active (prior to the interrupt).

Determining the thread/process responsible would be processor consuming, and may not provide the optimal assignment in all cases. Because interrupts tend to be short-lived events (relative to this time period) the last choice is quite acceptable.

The Microsoft kernels will follow the last option (although an accounting of interrupts is separately available) .

Task Manager

By default, once each second the Task Manager checks the counts of each task and computes the CPU utilization for that period. An example display is shown below. We have clicked on the CPU column header so that the list is sorted by CPU utilization.

The "System Idle" task is a task that runs whenever there is nothing else to do. Subtracting this value from 100 represents how busy the system is, which appears as the "CPU Usage" figure that appears at the bottom of the window.

This number is actually lower than the CPU Usage of the system in most cases. We have identified two causes for this.

Under-reporting due to truncation

The first clue that the numbers are not right come from the display itself. They always add exactly to 100.

When the Task Manager computes the CPU utilization of a given process, it displays an integer value. Thus, if the processes consumed 1.2% of the CPU it will display as 1%. The Task Manager uses truncation rather than rounding in this operation. 1.999% also appears as 1%. Thus the value shown for each process is usually a little lower than the actual value (but never above the actual value).

Because the numbers in the display always add to 100, the task manager is simply adding up these truncated values and subtracting from 100 to come up with the value for the System Idle Task.

Using kernel measurement techniques, we have confirmed that this is the case on the NT, 2000, and XP operating systems we have tested. We developed a non-truncating method of calculating the idle task that reveals a significantly better measurement than that of the task manager.

However, we found that there was more to consider...

Under-reporting due to interrupts

As mentioned earlier, when a time tick occurs during an interrupt, the Microsoft operating systems allocate the time to whatever task is running. If that happens to be the System Idle task, it is *effectively* allocated to it.

If we assume, for a minute, that interrupts are simply random events, then on a 10% loaded system 90% of the interrupts will occur in the System Idle task and will not show up in the CPU Usage figure of the Task Manager.

It turns out, however, that even on a 50% utilized system, more than 50% of the interrupts will occur during the system idle task. That is because interrupts are not simply random events - usually there is a task waiting on a queue that will be filled by the interrupt occurrence. Depending on the nature of the system, 60-90% of the interrupts can occur in the System Idle task of a 50% busy system.

These interrupts do represent real work being performed by the system, and they should be accounted for when considering how busy a system is.

Towards a better measurement

We developed a method of determining a more accurate indication of system CPU usage, that accounts for the under-reporting in the Task Manager discussed above. In some cases, the differences are minor, in the 1% range. On some systems, however, we have seen differences more more than 10% (of the CPU) at times. We attribute the worst case scenarios to relatively lightly loaded systems with poorly written device drivers. Some printer drivers in particular have a tendency to make the problem more visible.

A Disclaimer...

Quite honestly, we still have not accounted for some additional discrepancies that appear between our "better measurement" and the Task Manager under certain conditions. These discrepancies point to an additional under-reporting on the part of the Task Manager.

While DPCs (Delayed Procedure Calls) may ultimately be the source of additional under-reporting, current research does not support this theory. We feel that there is probably another source of error - possibly an error in accounting within the kernel. We'll keep looking.

[end]