White Paper: What's wrong with those 16-bit apps?

Why software uses 100% CPU

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

We have been hearing from a lot of users lately about problem "16-bit" applications. It seems that enterprises that have been upgrading to Windows 2000 Terminal Servers are finding these "legacy" applications incompatible with their new servers.

Here, we explain what is going on - and what you can do about it.

An Earlier Day

To understand what is happening, we have to remember how it was when these applications were developed. To most of us, a "server" in those days meant Unix! Windows was really a desktop operating system. These programs were designed in that environment.

The standard model of developing applications often included leftovers from the days of DOS. Tight software loops where the application would wait for something to happen. This eats up all the available CPU - but on a desktop that was OK.

Not all 16-bit applications cause these problems. It was an era of transition as new tools became available to the developers.

Modern software uses newer interfaces added to the operating system and standard libraries. These interfaces allow the application developer to wait for something by telling the OS to post a message to a queue when an event occurs. The application then waits on the queue, pausing the application until the event occurs (or an application defined timeout occurs). Using this method, the application uses no CPU resources while waiting.

Windows NT/TSE

Placing such an application on a multi-user server causes many problems. Even on a multi-CPU box, a single instance of such an application will max-out one of the CPUs, tying up not only that CPU but many other system busses and resources. Users see their keyboards freeze up (what we call limp-lock) and complain bitterly.

Windows NT and TSE included a utility that helped with this problem. Called KBDDOS, this utility could help contain 16 bit apps that performed a particularly bad form of polling. These applications typically were old DOS based apps that polled for the keyboard directly. The application would notice, for example, when the user pressed the shift or ctrl key. (Contrast this to today's applications that simply wait until a key sequence is complete.) Even when the application is waiting for the user and the user is away from the keyboard, the application manages to chew up all of the available CPU polling - just in case the user presses down on that alt-key!

Windows 2000 and above

Kernel changes in Windows 2000 made KBDDOS obsolete. Microsoft chose not to re-write it for Windows 2000. One can only conjecture on why they chose not to. Among the plausible answers would be:

- It wasn't as simple under Win2K.
- KBDDOS was a piecemeal answer. A more systematic approach would be better.
- They wanted to encourage that these old apps go away.

Enterprises upgrading to 2000 (or 2003) to avoid the end-of-life issues with NT are having trouble dealing with these applications. In most cases, these applications cannot be "modernized" because the original developer -- even the original code -- is long gone. The enterprise is left with either moving the application back to the desktop (which invalidates their investment in Terminal Servers) or totally replacing the application.

How to use TMuLimit to control these applications

TMuLimit is an operating system service that monitors and controls applications. This centralized approach is far more efficient than the old piecemeal approach of KBDDOS. TMuLimit uses a "Quality-of-Service" approach to ensure that all users and applications receive their fare-share of resources (and no more).

For most applications, TMuLimit uses a priority management approach. This allows reasonably written applications that require an occasional burst of significant processing power to use available spare CPU cycles when needed, but not at the expense of other users.

Many of these 16-bit applications, however, do not fall into the category of reasonable. Because the applications have such tight loops they don't give higher level priority applications a chance to run. For these applications, administrators configure TMuLimit to manage them using a hard CPU limit (what we call "SuperMax"). Most of the time, these applications can be configured to be prevented from using more than 10% of a single CPU and still provide the responsiveness that users expect.

Postponing the inevitable

Inevitably these applications will eventually be replaced in the enterprise someday. But replacing a key application is often much more than plunking down a new software package. People need to be trained, new processes and procedures need to be adopted. More often than not, this leads to a chain reaction to other changes and the scope of the project spirals out of control.

Better management of these applications, using a tool like TMuLimit, allow the enterprise to continue business as before immediately - without any big project to replace the known.

[end]