HYPACK® in a 64-bit World

By Bill Bergmann

Under the Hood (but not too far under)

The HYPACK® programming staff is busy developing 64-bit versions of the our software. Nearly all computers on the shelf at your electronics store of choice are sold with 64-bit processors inside and, with the advent of Windows 7, the Microsoft operating system supports this architecture as the new normal.

The good news is our current 32-bit HYPACK® software runs just fine in this environment. This is accomplished at the OS level through a special OS layer called WOW, an acronym for Windows On Windows. This layer handles the nitty-gritty details of converting on-the-fly differing pointer and integer sizes, allowing the processor and programs to interact without problems. One would imagine a price must be paid for this magic and they would be correct. It takes time to do this low-level translation from OS to program and back again. Native 64-bit programs don't require this intermediate step and thus are not subject to this particular time penalty.

Memory Pools

An even bigger benefit and the impetus for producing native 64-bit versions of our software is the ability to access really big chunks of memory. In 32-bit software, the amount of addressable memory, without resorting to various tricks, is limited to 32-bits. This translates to a theoretical 4 gigabytes total. Of this memory pool the first 2 are taken by the OS, so we are dropped to a 2 GB maximum. Again, due to various reasons, like pollution of dll loading at disparate addresses within the 2 GB program space, the real memory available to a program is considerably less than one would expect. It certainly does get used up fast.

The upshot of the above discussion for our programs is the quite real, hampering ability to process very large quantities of data, and if anything, the data sets are certainly getting larger. A 64-bit memory space is mind-bogglingly large and one which can easily accommodate today's and at least the not-too-distant future's data set sizes. In Windows® 7, each process can have a 16-terabyte virtual memory size.

The catch but a small one

What one must remember is that a 64-bit memory space, though incredibly large, is more concept than reality. It must still be backed up by physical memory chips. What happens when we allocate a gigabyte of memory in a program, but the computer has only 16 megabytes of physical memory? The OS in such cases will swap information to/from your hard drive and physical memory. This is a major reason your hard drive light is lit up so often. The OS is meeting the needs of the virtual vs physical memory requirements. This is beneficial for both the software and the user, in the sense that programmers can write simpler and cleaner code, which often means it is less likely to have bugs! It also lets the OS do the job it does best without the software getting in the way.

ONE SIZE FITS ALL?

As it turns out, even though 32-bit software can run under a 64-bit OS, the reverse is not true. Since there is a large base of 32-bit OS's out there, HYPACK® will be maintaining and releasing both bit versions of our software.

A point of consideration is whether to port a particular application to 64-bit at all, as the 32-bit version can run on either OS / processor. Some applications don't have demanding memory needs and porting them may have negligible benefits. We mentioned the time penalty of the WOW layer earlier but this can be offset by a decreased cache hit inside the processor. Suffice it to say that the processor can make predictions about where code is going and setup itself for a given path. If it guesses wrongly no harm is done, but it then must setup for the actual flow of the program. In 64-bits it is well known that cache misses are greater than under 32-bit.

Another point we consider in deciding what to port is the introduction of bugs. Unfortunately, it is not as simple as just recompiling code. It often requires changes to make our code work correctly in 64-bits, and changing code always opens the door to bug introduction.

WHICH VERSION AM I RUNNING?

For the curious, there are tools available, like 'P.E. Explorer', that can tell you a lot, probably more than you would ever care to know actually, about the internals of an application. Among this information would be if that program is a 32-bit or a 64-bit application / dynamic link library. However, there is a quick way to find this out and is already available in your Windows Task Manager. If you are running a 64-bit OS, 32-bit programs will show in the Task Manager with a '*32' appended to the process name. In the picture below I have circled to instances in red.

FIGURE 1. *Windows Task Manager Indicates 32- or 64-bit Programs*