Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Russell Ryan on Wed, 10 Jul 2013 02:57:13 GMT
View Forum Message <> Reply to Message

On Tuesday, July 9, 2013 5:47:44 PM UTC-4, Chip Helms wrote:
> Thanks, I hadn't seen that link. I'll have to take a closer look tomorrow. I got my actual project working using a while loop to wait for the child processes to finish. I have to say, I'm impressed by the lack of overhead. Using the four child processes resulted in the runtime for each iteration dropping from ~57s to ~15 seconds (my guess is it's because I only create the bridge objects once and then reuse them for each iteration...so the overhead is primarily passing data back and forth).
>
>
>
> Thanks again for the link. By the sound of it, I imagine it's a bit more elegant than what I've labeled as a babysitter loop.

Well, I'd start with mastering the IDL_IDLBridge before integrating it into a widget program.

But before you do that, let me strongly caution you. IDL_IDLBridge has a *MAJOR* bug in it --- there is a serious memory leak. IDL does not free the internal memory usage when an asychronous command. I'm certain of this behavior in Mac OSX, and have heard it's true for Linux (I don't actually have access to a Linux machine, but a friend said it was true). I don't use Windows, but think Exelis said it was not a problem on Windows. This AFFECTS parallelidl as well, because all that code does is all the book keeping of the bridges for you.

As with most things in life, there are work arounds. Unfortunately, this workaround requires you kill IDL and restart, so it's not really a work around per se. The memory leak does happen, but may not really kill you if you're only triggering each bridge a few dozen times. If you're triggering each bridge more than a few 100 times (and it doesn't seem to scale with the data operated on by the bridge), then you can start eating away at an appreciable amount of your RAM. If let run for a long time (many thousand triggers), it will completely kill the computer. The computer will start to swap to disk and the execution will grind to a halt.

Obviously, I've filed a bug report with Exelis. They confirmed everything I've said, and are "working" on it. That was many months (if not close to a year) ago, and it has not been addressed in any of the releases of IDL 8.x.

In short, I highly discourage you from using IDL_IDLBridge until this is fixed --- even if you're writing code that may not be affected (such as Windows) or expect the run to be short (so maybe the memory leaks won't pile up). Eventually, that code (if it's any good) will proliferate to workstations were it will cause problem. In my line of work, I'm aware of two IDL codes that have this problem and I see people happily using them. Normally I wouldn't care, but we share workstations and when they kill the machine, it hurts all of us.

But, if you must. Then yes. You need to poll every bridge and ask for its status. If the status is running, then go to the next bridge. If the bridge is done, then re-trigger. You'll find that a wait of short time between successive polls will actually be faster. Also, you'll need to be very careful because the order in which the bridges are started, may not be the order in which they finish. Consequently, whatever data the bridge processed may need to be sorted out, this can be accomplished by the USERDATA keyword to keep track of which data a bridge processed.

Good luck, you're going to need it.
Russell