
Subject: IDL performance, IDL_IDLBridge and multiple instances

Posted by [Laurent G.](#) on Fri, 18 Nov 2016 12:13:38 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

i'm doing number crunching in IDL 8.4 / Win7 / 64Bit and developed a simple IDL_IDLBridge construct to speed up a process. The process loads a single file, runs for about one hour while outputting log information, saves a new file and stops. I use 8 IDL_IDLBridge slaves to start 8 of these processes at the same time (8 core system). I poll every 10s with `bridge->status()` to check if a process is done. RAM is not a problem (10GB of 16GB available with all processes running). I used this technique before and got quasi linear speedups.

But this time the CPU load stays at 18%. Logging information tells me all 8 slave processes are running, but progressing very slowly. They seem to share a single core. Disc activity is very low and roughly corresponds to the 10KB logfile per hour and process.

All slaves run the same code, but with different parameters for input/output files. One of the main subroutines called contains a "common" block. It is first time i used that concept together with IDL_IDLBridge. The results look correct, so i guess this block is not cross-synchronized between slaves?

Is calling a subroutine limited to a single process/core?

Starting 3 instances of the IDL, each running 8 IDL_IDLBridge slaves also leads to ~18% CPU load. Again all 3x8 slaves are making progress, but very, very slowly, corresponding roughly to a single core shared by all 3x8 processes !?

Do those 3 IDL instances intercommunicate?

The problem happens on 2 windows 7 machines. Both machines run other IDL code just fine with all 8 cores loaded.

Thanks,
Laurent Gilson

Subject: Re: IDL performance, IDL_IDLBridge and multiple instances

Posted by [Laurent G.](#) on Fri, 18 Nov 2016 14:57:17 GMT

[View Forum Message](#) <> [Reply to Message](#)

Turns out common blocks are the problem: i used a common block to access a lookup hashtable in various places. Reworking the code to pass this hashtable into each method by reference solved the problem.
