Subject: Re: IDL and Dual Processor PC's
Posted by steinhh on Fri, 04 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <FCsyCq.22s@midway.uchicago.edu>
rivers@cars3.uchicago.edu (Mark Rivers) writes:

> In article <7j8714$631$1@readme.uio.no>, steinhh@ulrik.uio.no
>> (Stein Vidar Hagfors Haugan) writes:
[..]
>> ..then you could be getting a lot more value for money if you
>> buy two single-processor systems, running them in parallell
>> "by hand"...

> I might disagree with this. A dual-processor system, running 2
> instances of IDL simultaneously may be better value for money than a
> second complete computer. The incremental cost of the second CPU is
> not that high. One advantage is that you can put twice as much
> memory in this machine, and have all of it available to a single IDL
> process when you need it. This is what we are doing - 1 GB of
> memory on a dual-processor 450 MHz Pentium, Windows NT. We are
> running up against the 32 bit memory limitations of Windows and IDL.
> Even with 3 GB of swap space, IDL can only access arrays just over 1
> GB. It doesn't take a very big 3-D array to reach that limit!

Quite true, Mark. It's really a question that requires some thought on
how the system(s) are to be used, in addition to the actual prices per
CPU/board/memory/etc.

I just wanted to raise the issue, because I've often had a lot of
problems trying to explain to people the following scenario: You're
going to do some complex calculation 1000 times over, and then later
do statistics on the 1000 separate results. You want to think for a
while before running a process 1000 times using 10 processors in
parallell, rather than running a process 100 times on each of the
processors individually. Which way is faster depends a *lot* on the
problem at hand and on the machine architecture. With some problems
and some (memory-cached) architectures, splitting up the problem on 10
processors can mean you're done maybe 20 times faster! With some
problems, you might be unable to do it on a single machine anyway
(memory limitations). But for a wide range of applications, you're
getting the 1000 results faster by using good old sequential
processing.

And of course, if you're waiting for a single enhanced medical image
before you can start some life-saving operation after a car crash,
you'd want to go with the parallell version even if it doesn't give

you the most FLOPS/$.

Regards,

Stein Vidar