Subject: Re: When does IDL use MPs?
Posted by David Fanning on Wed, 13 Nov 2002 20:02:45 GMT
View Forum Message <> Reply to Message

Jonathan Greenberg (greenberg@ucdavis.edu) writes:

> Do I need to specifically invoke multiprocessor calls in an IDL procedure,
> or will it use MPs whenever appropriate?

Oh, oh. Those marketing guys have gotten out again. :-(

I'm afraid IDL is not multi-threaded. It can use
multiprocessors in one very narrowly defined situation:
when it is rendering object graphics volumes. Other
than that, you can invoke until the cows come home, but
it's not going to help much. Prayer would be more
effective, I think. :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: When does IDL use MPs?
Posted by mchinand on Wed, 13 Nov 2002 20:09:33 GMT
View Forum Message <> Reply to Message

In article <MPG.183c5dd4dd47868b989a17@news.frii.com>,
David Fanning <david@dfanning.com> wrote:
> Jonathan Greenberg (greenberg@ucdavis.edu) writes:
>
>> Do I need to specifically invoke multiprocessor calls in an IDL procedure,
>> or will it use MPs whenever appropriate?
>
> Oh, oh. Those marketing guys have gotten out again. :-(
>
> I'm afraid IDL is not multi-threaded. It can use
> multiprocessors in one very narrowly defined situation:
> when it is rendering object graphics volumes. Other
> than that, you can invoke until the cows come home, but
> it's not going to help much. Prayer would be more

> effective, I think. :-)
>
> Cheers,
>
> David
>

What about Chapter 2 in 'What's New in IDL 5.5':

'Muli-Threading in IDL'

Page 134 list about 2 dozen functions that support multi-threading.

This chapter also answers the original posters question. This file should
be the docs directory as a PDF file.

--Mike

Subject: Re: When does IDL use MPs?
Posted by David Fanning on Wed, 13 Nov 2002 20:21:24 GMT
View Forum Message <> Reply to Message

Mike Chinander (mchinand@midway.uchicago.edu) writes:

> What about Chapter 2 in 'What's New in IDL 5.5':
>
> 'Muli-Threading in IDL'
>
> Page 134 list about 2 dozen functions that support multi-threading.
>
> This chapter also answers the original posters question. This file should
> be the docs directory as a PDF file.

Well, there you go. I'm beginning to understand how
Craig feels, as I seem to be completely stuck in IDL 5.4. :-(

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: When does IDL use MPs?
Posted by Craig Markwardt on Thu, 14 Nov 2002 04:21:06 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> writes:
>
> Well, there you go. I'm beginning to understand how
> Craig feels, as I seem to be completely stuck in IDL 5.4. :-(

Umm, but I'm stuck in IDL *5.2* You have no idea how I feel :-)

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: When does IDL use MPs?
Posted by JD Smith on Fri, 15 Nov 2002 22:20:08 GMT
View Forum Message <> Reply to Message

On Wed, 13 Nov 2002 13:02:45 -0700, David Fanning wrote:

> Jonathan Greenberg (greenberg@ucdavis.edu) writes:
>
>> Do I need to specifically invoke multiprocessor calls in an IDL
>> procedure, or will it use MPs whenever appropriate?
>
> Oh, oh. Those marketing guys have gotten out again. :-(
>
> I'm afraid IDL is not multi-threaded. It can use multiprocessors in one
> very narrowly defined situation: when it is rendering object graphics
> volumes. Other than that, you can invoke until the cows come home, but
> it's not going to help much. Prayer would be more effective, I think.
> :-)
>

Well, it's slightly better than that. IDL is designed to use its
thread pool for a number of the most important operations and routines
(WHERE, FFT, basic array arithmetic, etc.) on MP machines whenever
appropriate. The bad news is that the overhead for spawning a
separate thread to compute, say, the total of a big array is
significant, motivating the default choice for the minimum number of
elements before threading occurs to be large:

```
help,!CPU,/STRUCTURES
** Structure !CPU, 6 tags, length=24, data length=24:
   HW_VECTOR       LONG                 0
   VECTOR_ENABLE   LONG                 0
   HW_NCPU         LONG                 2
   TPOOL_NTHREADS  LONG                 2
   TPOOL_MIN_ELTS  LONG            100000
   TPOOL_MAX_ELTS  LONG                 0
```

I.e. unless you're operating on an array with more than 100,000 elements, you are out of luck (TPOOOL_MIN_ELTS). You can change this value globally with the CPU procedure, or hand tune it in the call to any of the 50 or so routines which are threaded, but 100,000 is usually about right.

When the magic is right, it really cooks on my dual-processor system:

```
IDL> r=randomu(sd,10000000L)
IDL> t=systime(1) & tot=total(r,TPOOL_MIN_ELTS=10000001L) & print,systime(1)-t
      0.11507905
IDL> t=systime(1) & tot=total(r,TPOOL_MIN_ELTS=10000000L) & print,systime(1)-t
      0.061390042
```

I find measurable speedups even for 50,000 elements, but it really shines on much larger chunks of data. It also depends on the function, of course. Experiment and see what you find.

JD
