Subject: Parallel Processing in IDL
Posted by Ammar Yusuf on Mon, 11 Oct 2010 16:35:16 GMT
View Forum Message <> Reply to Message

What's an easy way to use multiple processors in IDL? I have a large program but I want to start with a simple program first. Let's say I have an array of a million integers and I have four processors. I want to add all the elements in this array and output it. I could split the array into four parts and give it to each processor right? Is this possible? How would I do this? Thanks!

Subject: Re: Parallel Processing in IDL
Posted by natha on Tue, 12 Oct 2010 13:30:30 GMT
View Forum Message <> Reply to Message

On Oct 12, 8:34 am, chris <rog...@googlemail.com> wrote:
> On 12 Okt., 00:13, nata <bernat.puigdomen...@gmail.com> wrote:
>
>
>
>> On 11 oct, 12:35, Ammar Yusuf <amyus...@gmail.com> wrote:
>
>>> What's an easy way to use multiple processors in IDL? I have a large
>>> program but I want to start with a simple program first. Let's say I
>>> have an array of a million integers and I have four processors. I want
>>> to add all the elements in this array and output it. I could split the
>>> array into four parts and give it to each processor right? Is this
>>> possible? How would I do this? Thanks!
>
>> I wrote a routine to use multi-threading some time ago. I use
>> IDL_IDLBridge and SHMMAP and it works good for some cases. If you are
>> interested I can send you a copy.
>> It only works for functions without output keywords.
>
>> Cheers,
>> nata
>
> Hi Nata,
> please send the copy also to me :)
>
> Cheers
>
> CR

I will send you a copy this evening, I have to check the versions I have.
Have a nice day,

nata

Subject: Re: Parallel Processing in IDL
Posted by Giuseppe Papa on Sat, 26 Feb 2011 03:18:38 GMT
View Forum Message <> Reply to Message

freelance writer

Subject: Re: Parallel Processing
Posted by Russell Ryan on Thu, 28 Jun 2012 14:27:47 GMT
View Forum Message <> Reply to Message

On Thursday, June 28, 2012 10:05:24 AM UTC-4, stefan....@gmail.com wrote:
> Hi
>
> I have developed a code which takes a couple of hours to run and I am aware of the fact that IDL automatically parallelizes some vector operations and one should prefer those instead of looping through arrays.
>
> I have done all that but still I know I could speed up things by a factor of 2 when I do certain things on 2 cores.
>
> For instance, somewhere in the program I pass some arrays to a function and this function then returns and equally large array with some calculated values. This is all done with one core since the operations in the function are not parallelized.
>
> However, I could split up the input arrays into to equally large parts and perform the calcualtions for each of those two on one core. In the end, when both are finished I could just concatenate the result-arrays.
>
> Is this possible in some easy way?
>
> thanks for your help :)

Hi Stefan,
I know of at least three ways to skin this cat, and they depend on the details of your problem and how much money you're willing to spend.

(1) Simply break the problem into several smaller bits, and run each bit in different IDL sessions. The OS will naturally divvy the computations accordingly. Of course, this is the easiest, but may not be practical for your problem.

(2) Employ the IDL_IDLBridge architecture built by ITT for IDL. This adds a new level of programming, but it's a way of running multiple IDL commands simultaneously. At first glance,

this sounds tedious --- but I"ve done it 1000s of times and it's not too bad. This works best for problems where you want to preform some long task on one set of data and then go to a next set --- and the sets are unrelated. For example, have 100 objects and each object must be processed identically.

(3) There is some aftermarket software (I believe it's called FastDL) that you can build to do this. I'm not to familiar with it, but you can google it.

Russell

Subject: Re: Parallel Processing
Posted by stefan.meingast on Thu, 28 Jun 2012 14:35:05 GMT
View Forum Message <> Reply to Message

Am Donnerstag, 28. Juni 2012 16:27:47 UTC+2 schrieb Russell:
> On Thursday, June 28, 2012 10:05:24 AM UTC-4, stefan....@gmail.com wrote:
>> Hi
>>
>> I have developed a code which takes a couple of hours to run and I am aware of the fact that IDL automatically parallelizes some vector operations and one should prefer those instead of looping through arrays.
>>
>> I have done all that but still I know I could speed up things by a factor of 2 when I do certain things on 2 cores.
>>
>> For instance, somewhere in the program I pass some arrays to a function and this function then returns and equally large array with some calculated values. This is all done with one core since the operations in the function are not parallelized.
>>
>> However, I could split up the input arrays into to equally large parts and perform the calcualtions for each of those two on one core. In the end, when both are finished I could just concatenate the result-arrays.
>>
>> Is this possible in some easy way?
>>
>> thanks for your help :)
>
>
> Hi Stefan,
> I know of at least three ways to skin this cat, and they depend on the details of your problem and how much money you're willing to spend.
>
> (1) Simply break the problem into several smaller bits, and run each bit in different IDL sessions. The OS will naturally divvy the computations accordingly. Of course, this is the easiest, but may not be practical for your problem.
>
> (2) Employ the IDL_IDLBridge architecture built by ITT for IDL. This adds a new level of

programming, but it's a way of running multiple IDL commands simultaneously. At first glance, this sounds tedious --- but I"ve done it 1000s of times and it's not too bad. This works best for problems where you want to preform some long task on one set of data and then go to a next set --- and the sets are unrelated. For example, have 100 objects and each object must be processed identically.
>
> (3) There is some aftermarket software (I believe it's called FastDL) that you can build to do this. I'm not to familiar with it, but you can google it.
>
> Russell

Thanks for your fast reply. I was think of splitting everything up into two parts by just starting two sessions separately with different job assignments and then in the end just merge the results. For me this approach is not attractive since a) I have to set up two different initial sets each time I run something and b) I don't learn anything about parallel processing.

I think I will look into the IDLbridge and see whats in there

thanks
cheers

Subject: Re: Parallel Processing
Posted by Russell Ryan on Thu, 28 Jun 2012 16:40:15 GMT
View Forum Message <> Reply to Message

On Thursday, June 28, 2012 10:05:24 AM UTC-4, stefan....@gmail.com wrote:
> Hi
>
> I have developed a code which takes a couple of hours to run and I am aware of the fact that IDL automatically parallelizes some vector operations and one should prefer those instead of looping through arrays.
>
> I have done all that but still I know I could speed up things by a factor of 2 when I do certain things on 2 cores.
>
> For instance, somewhere in the program I pass some arrays to a function and this function then returns and equally large array with some calculated values. This is all done with one core since the operations in the function are not parallelized.
>
> However, I could split up the input arrays into to equally large parts and perform the calcualtions for each of those two on one core. In the end, when both are finished I could just concatenate the result-arrays.
>
> Is this possible in some easy way?
>
> thanks for your help :)

Yeah, that sounds like it's what you want. Post back if it's not clear how to proceed. I use this stuff all the time for several big pipelines that I use, but it only works if you've got a relatively small amount of data to process (which may take a long time). Then you want repeat this procedure for many similar units of data.

Russell

Subject: Re: Parallel Processing
Posted by stefan.meingast on Thu, 28 Jun 2012 17:02:51 GMT
View Forum Message <> Reply to Message

Am Donnerstag, 28. Juni 2012 18:40:15 UTC+2 schrieb (unbekannt):
> On Thursday, June 28, 2012 10:05:24 AM UTC-4, stefan....@gmail.com wrote:
>> Hi
>>
>> I have developed a code which takes a couple of hours to run and I am aware of the fact that IDL automatically parallelizes some vector operations and one should prefer those instead of looping through arrays.
>>
>> I have done all that but still I know I could speed up things by a factor of 2 when I do certain things on 2 cores.
>>
>> For instance, somewhere in the program I pass some arrays to a function and this function then returns and equally large array with some calculated values. This is all done with one core since the operations in the function are not parallelized.
>>
>> However, I could split up the input arrays into to equally large parts and perform the calcualtions for each of those two on one core. In the end, when both are finished I could just concatenate the result-arrays.
>>
>> Is this possible in some easy way?
>>
>> thanks for your help :)
>
> Yeah, that sounds like it's what you want. Post back if it's not clear how to proceed. I use this stuff all the time for several big pipelines that I use, but it only works if you've got a relatively small amount of data to process (which may take a long time). Then you want repeat this procedure for many similar units of data.
>
> Russell

Hey

It works but I had to take a step back since I ran into some quite ridiculous problem...

It actually takes longer for IDL to set up, say, two child processes and then run them on separate cores as compared to running through 2 iterations on one core.

So I implemented the IDLbridge right at the beginning and separated the whole process into 2 substeps and not just this one step where I have to do one calculation.

If someone else should read this in the future:
It took me about 10 minutes to set up my first code to run on 2 cores without knowing ANYTHING about that stuff.

Here is what guided me through:
http://slugidl.pbworks.com/w/page/29199259/Child%20Processes

very easy
cheers
