Subject: Parallel Processing
Posted by stefan.meingast on Thu, 28 Jun 2012 14:05:24 GMT
View Forum Message <> Reply to Message

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 :)

Subject: Re: Parallel Processing
Posted by stefan.meingast on Thu, 05 Jul 2012 14:50:38 GMT
View Forum Message <> Reply to Message

Hi

Now that I have successfully implemented multi-threading another prblem occured:

To invoke multiple processes is start in a loop with

bridges[i]->EXECUTE, 'program,par1,par2', /NOWAIT

where bridges is an object-array which holds the different child processes.
Upon the execution of the last process I do

bridges[i]->EXECUTE, 'program,par3,par4'

And after that I destroy my bridges in a loop.

Now I have a problem if the last process finishes before one of the previous since upon its completion it will directly move to the part where all bridges are destroyed and kills my program...

Is there an easy way to tell IDL to wait for all my processes to finish and then destroy the bridges?

thanks
Stefan

---

## Subject: Re: Parallel Processing
Posted by stefan.meingast on Thu, 05 Jul 2012 14:52:23 GMT
View Forum Message <> Reply to Message

Hi

Now that I have successfully implemented multi-threading another problem occured:

To invoke multiple processes I start in a loop with

bridges[i]->EXECUTE, 'program,par1,par2', /NOWAIT

where 'bridges' is an object-array which holds the different child processes.
Upon the execution of the last process I do

bridges[i]->EXECUTE, 'program,par3,par4'

And after that I destroy my bridges in a loop.

Now I have a problem if the last process finishes before one of the previous processes since upon its completion it will directly move to the part where all bridges are destroyed and kills my program...

Is there an easy way to tell IDL to wait for all my processes to finish and then destroy the bridges?

thanks
Stefan

---

## Subject: Re: Parallel Processing
Posted by Lajos Foldy on Thu, 05 Jul 2012 15:13:02 GMT
View Forum Message <> Reply to Message

On Thursday, July 5, 2012 4:52:23 PM UTC+2, Stefan wrote:
> Hi
>
> Now that I have successfully implemented multi-threading another problem occured:
>
> To invoke multiple processes I start in a loop with
>

> bridges[i]->EXECUTE, 'program,par1,par2', /NOWAIT
>
> where 'bridges' is an object-array which holds the different child processes.
> Upon the execution of the last process I do
>
> bridges[i]->EXECUTE, 'program,par3,par4'
>
> And after that I destroy my bridges in a loop.
>
> Now I have a problem if the last process finishes before one of the previous processes since upon its completion it will directly move to the part where all bridges are destroyed and kills my program...
>
> Is there an easy way to tell IDL to wait for all my processes to finish and then destroy the bridges?
>
> thanks
> Stefan

Use IDL_IDLBridge::Status to determine whether the job is finished. Pseudocode:

```
ndone=0
running=replicate(1, njobs)
while ndone lt njobs begin
   for j=0,njobs-1 do begin
      if running[j] then begin
        query status of the j-th job with IDL_IDLBridge::Status
        if finished
          destroy bridge
          running[j]=0
          ndone++
        endif
      endif
   endfor
   wait, 1
endwhile
```

regards,
Lajos

Subject: Re: Parallel Processing
Posted by stefan.meingast on Thu, 05 Jul 2012 15:16:27 GMT
View Forum Message <> Reply to Message

Am Donnerstag, 5. Juli 2012 17:13:02 UTC+2 schrieb fawltyl...@gmail.com:
> On Thursday, July 5, 2012 4:52:23 PM UTC+2, Stefan wrote:

>> Hi
>>
>> Now that I have successfully implemented multi-threading another problem occured:
>>
>> To invoke multiple processes I start in a loop with
>>
>> bridges[i]->EXECUTE, 'program,par1,par2', /NOWAIT
>>
>> where 'bridges' is an object-array which holds the different child processes.
>> Upon the execution of the last process I do
>>
>> bridges[i]->EXECUTE, 'program,par3,par4'
>>
>> And after that I destroy my bridges in a loop.
>>
>> Now I have a problem if the last process finishes before one of the previous processes since upon its completion it will directly move to the part where all bridges are destroyed and kills my program...
>>
>> Is there an easy way to tell IDL to wait for all my processes to finish and then destroy the bridges?
>>
>> thanks
>> Stefan
>
> Use IDL_IDLBridge::Status to determine whether the job is finished. Pseudocode:
>
```
> ndone=0
> running=replicate(1, njobs)
> while ndone lt njobs begin
>    for j=0,njobs-1 do begin
>       if running[j] then begin
>          query status of the j-th job with IDL_IDLBridge::Status
>          if finished
>             destroy bridge
>             running[j]=0
>             ndone++
>          endif
>       endif
>    endfor
>    wait, 1
> endwhile
```
>
>
> regards,
> Lajos

Ah, I see you put WAIT there if its not finished. I thought of using this in a loop but without putting

WAIT you spend too many resources.

Thanks! :)

---

Subject: Re: Parallel Processing
Posted by Russell Ryan on Thu, 05 Jul 2012 15:29:21 GMT
View Forum Message <> Reply to Message

On Thursday, July 5, 2012 11:16:27 AM UTC-4, Stefan wrote:
> Am Donnerstag, 5. Juli 2012 17:13:02 UTC+2 schrieb fawltyl...@gmail.com:
>> On Thursday, July 5, 2012 4:52:23 PM UTC+2, Stefan wrote:
>>> Hi
>>>
>>> Now that I have successfully implemented multi-threading another problem occured:
>>>
>>> To invoke multiple processes I start in a loop with
>>>
>>> bridges[i]->EXECUTE, 'program,par1,par2', /NOWAIT
>>>
>>> where 'bridges' is an object-array which holds the different child processes.
>>> Upon the execution of the last process I do
>>>
>>> bridges[i]->EXECUTE, 'program,par3,par4'
>>>
>>> And after that I destroy my bridges in a loop.
>>>
>>> Now I have a problem if the last process finishes before one of the previous processes since upon its completion it will directly move to the part where all bridges are destroyed and kills my program...
>>>
>>> Is there an easy way to tell IDL to wait for all my processes to finish and then destroy the bridges?
>>>
>>> thanks
>>> Stefan
>>
>> Use IDL_IDLBridge::Status to determine whether the job is finished. Pseudocode:
>>

```
>> ndone=0
>> running=replicate(1, njobs)
>> while ndone lt njobs begin
>>    for j=0,njobs-1 do begin
>>       if running[j] then begin
>>          query status of the j-th job with IDL_IDLBridge::Status
>>          if finished
>>             destroy bridge
>>             running[j]=0
>>              ndone++
>>           endif
>>        endif
>>     endfor
>>     wait, 1
>>  endwhile
>>
>>
>>  regards,
>>  Lajos
>
```
> Ah, I see you put WAIT there if its not finished. I thought of using this in a loop but without putting WAIT you spend too many resources.
>
> Thanks! :)

Yeah, I put a wait in there too.  I've thought about "calibrating" the wait time.  It seems to me that if your process takes (say 30 minutes) then polling the bridges every 1 second seems like overkill. My gut-feeling is that this wait time should be roughly 1/3 or 1/4 the expected time per process, but of course you'd want to ensure that it's always waiting a minimum of ~0.5 s.

Russell

---

## Subject: Re: Parallel Processing
## Posted by lecacheux.alain on Thu, 05 Jul 2012 15:53:26 GMT

View Forum Message <> Reply to Message

On 5 juil, 17:16, Stefan <stefan.meing...@gmail.com> wrote:
> Am Donnerstag, 5. Juli 2012 17:13:02 UTC+2 schrieb fawltyl...@gmail.com:
>
>
>
>
>
>> On Thursday, July 5, 2012 4:52:23 PM UTC+2, Stefan wrote:
>>> Hi
>
>>> Now that I have successfully implemented multi-threading another problem occured:
>
>>> To invoke multiple processes I start in a loop with
>
>>> bridges[i]->EXECUTE, 'program,par1,par2', /NOWAIT
>
>>> where 'bridges' is an object-array which holds the different child processes.
>>> Upon the execution of the last process I do
>
>>> bridges[i]->EXECUTE, 'program,par3,par4'

>
>>>  And after that I destroy my bridges in a loop.
>
>>>  Now I have a problem if the last process finishes before one of the previous processes since upon its completion it will directly move to the part where all bridges are destroyed and kills my program...
>
>>>  Is there an easy way to tell IDL to wait for all my processes to finish and then destroy the bridges?
>
>>>  thanks
>>>  Stefan
>
>>  Use IDL_IDLBridge::Status to determine whether the job is finished. Pseudocode:
>

```
>>  ndone=0
>>  running=replicate(1, njobs)
>>  while ndone lt njobs begin
>>     for j=0,njobs-1 do begin
>>        if running[j] then begin
>>          query status of the j-th job with IDL_IDLBridge::Status
>>          if finished
>>             destroy bridge
>>             running[j]=0
>>             ndone++
>>          endif
>>        endif
>>     endfor
>>     wait, 1
>>  endwhile
```

>
>>  regards,
>>  Lajos
>
>  Ah, I see you put WAIT there if its not finished. I thought of using this in a loop but without putting WAIT you spend too many resources.
>
>  Thanks! :)
>
>

My way to manage that is putting /NOWAIT, but waiting inside the loop, as follows:

```
  repeat begin
    wait, 1
    test = 1B
    foreach b,bridges do test AND= (b.Status() ne 1)
```

endrep until test

alain.