Subject: Re: IDL with multiple processors
Posted by korpela on Fri, 04 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article <748sgnrck1@readme.uio.no>,
Stein Vidar Hagfors Haugan <steinhh@ulrik.uio.no> wrote:
>> It's also probably illegal to use fork on a machine with
>> a floating license.
>
> Actually, I don't think RSI would mind from the licensing point of
> view: AFAIK, they have a "one user, one screen, any number of processes"
> policy (you can even run two IDL processes on separate machines,
> but on the same display, using just one license).

I'm pretty sure that when using floating licences, every IDL process I start checks out 10 licences regardless of which display they are running on. It's only the static licenses that allow any number of processes to be started as long as they are running on the same machine (regardless of the display location).

Eric
--
Eric Korpela | An object at rest can never be
korpela@ssl.berkeley.edu | stopped.
Click for home page.

Subject: Re: IDL with multiple processors
Posted by thompson on Fri, 04 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

steinhh@ulrik.uio.no (Stein Vidar Hagfors Haugan) writes:

> In article <7479pd$j5i$1@agate.berkeley.edu>
> korpela@islay.ssl.berkeley.edu (Eric J. Korpela) writes:

>> IDL is pretty much single threaded. If you've got two processors, it's up
>> to you to use 'em. You'd be suprised what you can do if you try.....
>> This works under sunos... (you need to write your own kill proceedure,
>> though. That's not too hard.)
>>
>> pid=call_external("/usr/lib/libc.so.1.9","_fork") ; your libc name may vary
>> if pid then begin
>> do some processing
>> kill,pid
>> endif else begin
>> do some other processing

>>   dummy=call_external("/usr/lib/libc.so.1.9","_wait")
>> endelse
>>

> Hmmm... dreaming of a "PIDL" (Parallel IDL).... How about
> something like a few extra "directives":

 (rest deleted)

I always thought of IDL as an ideal candidate for parallelization, without any need for modified code. After all, one can apply an operation to an entire array of data in one call, i.e. if one says

 A = B + C

and B and C are arrays, then this is equivalent to N separate operations on each of the elements of the arrays. Why couldn't this be split up over several processors?

William Thompson

Subject: Re: IDL with multiple processors
Posted by steinhh on Fri, 04 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article <7479pd$j5i$1@agate.berkeley.edu>
korpela@islay.ssl.berkeley.edu (Eric J. Korpela) writes:

> IDL is pretty much single threaded. If you've got two processors, it's up
> to you to use 'em. You'd be suprised what you can do if you try.....
> This works under sunos... (you need to write your own kill proceedure,
> though. That's not too hard.)
>
> pid=call_external("/usr/lib/libc.so.1.9","_fork") ; your libc name may vary
> if pid then begin
>   do some processing
>   kill,pid
> endif else begin
>   do some other processing
>   dummy=call_external("/usr/lib/libc.so.1.9","_wait")
> endelse
>

Hmmm... dreaming of a "PIDL" (Parallel IDL).... How about
something like a few extra "directives":

  parallel begin

```
    task begin
      <some processing>
    endtask

    task begin
      <some other processing>
    endtask

    task begin
      <some third kind of processing>
    endtask

  endpara
```

... having approximately the same meaning as your example,
but with an unspecified mechanism. If no parallel processing
is available on the platform, IDL could just evaluate the tasks
in sequence. I.e. the programmer should never *rely* on these
tasks being executed in parallel, and (for now) interprocess
communication should thus not be used.. but making the
mechanism more or less like threads (threads share the
same address space/global variables) would eliminate most
of the need - synchronization would be done by syntax instead
(at each "endpara" directive).

In fact, RSI could implement something like this pretty soon
(after a tiny period of thinking it through, but without
putting any meat on the bone until later... Just
allowing the directives and ignoring them would be ok,
but one would at least be able to write programs that
will be speeded up in some future version...

I don't see it happening, though in my view it could be a nice
selling point over competitors to have "mechanisms to take
advantage of multiple processors, anticpating future h/w & s/w
developments".

[..snip..]

> I don't suppose RSI would be willing to add fork, kill, and wait to the
> language. :) It's also probably illegal to use fork on a machine with
> a floating license.

Actually, I don't think RSI would mind from the licensing point of
view: AFAIK, they have a "one user, one screen, any number of processes"
policy (you can even run two IDL processes on separate machines,
but on the same display, using just one license).

Regards,

Stein Vidar

Subject: Re: IDL with multiple processors
Posted by korpela on Fri, 04 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article <3667038C.1474744F@lanl.gov>,
David M. Schmidt <dschmidt@lanl.gov> wrote:
> We recently tested the speed of a non-graphical, numerical IDL code run
> on two different Linux systems. The system with 2 350 MHz processors
> and SDRAM was about the same speed or a bit slower than the system with
> 1 233 MHz processor and EDORAM.

IDL is pretty much single threaded. If you've got two processors, it's up
to you to use 'em. You'd be suprised what you can do if you try.....
This works under sunos... (you need to write your own kill proceedure,
though. That's not too hard.)

```
pid=call_external("/usr/lib/libc.so.1.9","_fork") ; your libc name may vary
if pid then begin
  do some processing
  kill,pid
endif else begin
  do some other processing
  dummy=call_external("/usr/lib/libc.so.1.9","_wait")
endelse
```

You can even use the "varray" package at my web site for interprocess
communication. (Otherwise changes to variables in one process do not
affect the other process.)

I don't suppose RSI would be willing to add fork, kill, and wait to the
language. :) It's also probably illegal to use fork on a machine with
a floating license.

Eric

--
Eric Korpela | An object at rest can never be
korpela@ssl.berkeley.edu | stopped.
Click for home page.

Subject: Re: IDL with multiple processors
Posted by steinhh on Sat, 05 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article <749m0g$875$1@agate.berkeley.edu>
korpela@islay.ssl.berkeley.edu (Eric J. Korpela) writes:

> I'm pretty sure that when using floating licences, every IDL process I
> start checks out 10 licences regardless of which display they are running
> on. It's only the static licenses that allow any number of processes to
> be started as long as they are running on the same machine (regardless
> of the display location).

A little experimenting shows that starting an extra idl process
on the same machine, with the same display, does not allocate
extra licenses. With three idl processes on the machine,
lmstat -A reports only 10 licenses in use. Starting a
process on another machine (regardless of display destination)
requires another 10, but additional processes on that
machine (with the same display) requires no more licenses).

Stein Vidar

Subject: Re: IDL with multiple processors
Posted by steinhh on Sat, 05 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article <749919$okv@post.gsfc.nasa.gov>
thompson@orpheus.nascom.nasa.gov (William Thompson) writes:

> I always thought of IDL as an ideal candidate for parallelization,
> without any need for modified code. After all, one can apply an
> operation to an entire array of data in one call, i.e. if one says
>
> A = B + C
>
> and B and C are arrays, then this is equivalent to N separate
> operations on each of the elements of the arrays. Why couldn't
> this be split up over several processors?

You are of course right, Bill. This is definitely the optimal
strategy for parallelization for (well-written) IDL applications.
Among other things, it definitely scales a lot better with increasing
number of processors than writing a few threads that may execute
in parallell.

The only thing I can say to my defence for not mentioning it

is that it's, well, kind of boring, from a programmers perspective :-)

Stein Vidar

Subject: Re: IDL with multiple processors
Posted by David Schmidt on Mon, 07 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

William Thompson wrote:
> I always thought of IDL as an ideal candidate for parallelization, without any
> need for modified code. After all, one can apply an operation to an entire
> array of data in one call, i.e. if one says
>
> A = B + C
>
> and B and C are arrays, then this is equivalent to N separate operations on
> each of the elements of the arrays. Why couldn't this be split up over several
> processors?
--

Does anyone know if RSI has this or any other (e.g. multi-thread)
multi-processor enhancements in the works?

David
