Subject: Re: Pass by value and performance
Posted by peter.albert@gmx.de on Wed, 14 Dec 2005 07:38:05 GMT
View Forum Message <> Reply to Message

Hi Ken,

if you used pointers in your structures and routines and only passed
the pointers, your code could stay nice and you would save a lot of
memory.

Cheers,

Peter

Subject: Re: Pass by value and performance
Posted by Kenneth P. Bowman on Wed, 14 Dec 2005 15:09:19 GMT
View Forum Message <> Reply to Message

In article <1134545885.801135.229150@g49g2000cwa.googlegroups.com>,
"Peter Albert" <peter.albert@gmx.de> wrote:

> Hi Ken,
>
> if you used pointers in your structures and routines and only passed
> the pointers, your code could stay nice and you would save a lot of
> memory.
>
> Cheers,
>
> Peter

True, and I considered that possibility. It would work fine for passing
the arrays amongst my own routines, but I have to dereference the
pointer to pass the array itself into system routines. Dereferencing a
pointer is an expression, is it not?, and expressions are passed by
value.

Ken

Subject: Re: Pass by value and performance
Posted by David Fanning on Wed, 14 Dec 2005 15:53:08 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

> True, and I considered that possibility. It would work fine for passing
> the arrays amongst my own routines, but I have to dereference the
> pointer to pass the array itself into system routines. Dereferencing a
> pointer is an expression, is it not?, and expressions are passed by
> value.

Ah, you would be interested to read the new pointer
section of my book that I just completely yesterday!
(Well, actually, you could read the damn pointer tutorial
where I learned about this, too.) Here is the news that
even I didn't fully appreciate until I put it down in my
own words. Pointers are just regular IDL variables!!

Consider this function:

```
PRO JUNK, var
  var = var * 5
END
```

And do this:

```
IDL> a = Ptr_New(5)
IDL> junk, *a
```

Here is the question. What is the result of this command?

```
IDL> Print, *a
```

The answer might surprise you! :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Pass by value and performance
Posted by Paul Van Delst[1] on Wed, 14 Dec 2005 17:02:14 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Kenneth P. Bowman writes:
>
>

>> True, and I considered that possibility. It would work fine for passing
>> the arrays amongst my own routines, but I have to dereference the
>> pointer to pass the array itself into system routines. Dereferencing a
>> pointer is an expression, is it not?, and expressions are passed by
>> value.
>
>
> Ah, you would be interested to read the new pointer
> section of my book that I just completely yesterday!
> (Well, actually, you could read the damn pointer tutorial
> where I learned about this, too.) Here is the news that
> even I didn't fully appreciate until I put it down in my
> own words. Pointers are just regular IDL variables!!

In the context of this thread (and your example below), shouldn't one say that
/dereferenced/ pointers are just regular IDL variables (i.e. they're not expressions) ??

I don't think I'm splitting hairs here (or am I?)

>
> Consider this function:
>
>   PRO JUNK, var
>   var = var * 5
>   END
>
> And do this:
>
>   IDL> a = Ptr_New(5)
>   IDL> junk, *a
>
> Here is the question. What is the result of this command?
>
>   IDL> Print, *a
>
> The answer might surprise you! :-)

After my recent grokking of objects (thanks to that "object for dummies" website posted
earlier this month), the answer didn't surprise me as much it would of, but it's still a
good reinforcement.

It makes me wonder though - have IDL pointers always behaved that way, or is it a moving
target (ala relaxed structure definitions post IDL 5.3 talked about in another thread) ?

paulv

--
Paul van Delst

Subject: Re: Pass by value and performance
Posted by David Fanning on Wed, 14 Dec 2005 17:08:30 GMT
View Forum Message <> Reply to Message

Paul Van Delst writes:

> In the context of this thread (and your example below), shouldn't one say that
> /dereferenced/ pointers are just regular IDL variables (i.e. they're not expressions) ??
>
> I don't think I'm splitting hairs here (or am I?)

No, I think that is what I meant. I was just so darned
excited about knowing *why* this answer was true I mis-spoke. :-)

> It makes me wonder though - have IDL pointers always behaved that way, or is it a moving
> target (ala relaxed structure definitions post IDL 5.3 talked about in another thread) ?

I think they have always behaved this way. But often these things
are introduced without, uh, too much fanfare, and it takes everyone
a while to catch up.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Pass by value and performance
Posted by Kenneth P. Bowman on Wed, 14 Dec 2005 17:28:07 GMT
View Forum Message <> Reply to Message

Perhaps someone can clarify this for me.

I was doing this

```
data = {values : FLTARR(...), $
        other  : other stuff ...}
```

Then pass "data" to a procedure and do this

result = INTERPOLATE(data.values, x, y, z)

In this case data.values is passed by reference. (p. 92 of Building IDL Applications)

If I were to use pointers, I could do something like this instead

data = {values : PTR_NEW(FLTARR(...)), $
        other : other stuff ...}

Pass "data" to a procedure and do this

result = INTERPOLATE(*data.values, x, y, z)

Is *data.values passed by reference or by value? And how does one tell?

Ken

Subject: Re: Pass by value and performance
Posted by David Fanning on Wed, 14 Dec 2005 17:54:31 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

> Perhaps someone can clarify this for me.
>
> I was doing this
>
> data = {values : FLTARR(...), $
>   other : other stuff ...}
>
> Then pass "data" to a procedure and do this
>
> result = INTERPOLATE(data.values, x, y, z)
>
> In this case data.values is passed by reference. (p. 92 of Building IDL
> Applications)
>
>
> If I were to use pointers, I could do something like this instead
>
> data = {values : PTR_NEW(FLTARR(...)), $
>   other : other stuff ...}
>
> Pass "data" to a procedure and do this

>
> result = INTERPOLATE(*data.values, x, y, z)
>
> Is *data.values passed by reference or by value? And how does one tell?

I would test it with my little JUNK program from before,
like so:

```
IDL> b = {ptr:Ptr_New(5)}
IDL> junk, *b.ptr
IDL> help, *b.ptr
<PtrHeapVar26>  INT       =       25
```

Whoa! Excuse me. I need to do a little more work on that
pointer chapter.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Pass by value and performance
Posted by JD Smith on Wed, 14 Dec 2005 19:04:10 GMT
View Forum Message <> Reply to Message

On Wed, 14 Dec 2005 10:08:30 -0700, David Fanning wrote:

> Paul Van Delst writes:
>
>> In the context of this thread (and your example below), shouldn't one
>> say that /dereferenced/ pointers are just regular IDL variables (i.e.
>> they're not expressions) ??
>>
>> I don't think I'm splitting hairs here (or am I?)
>
> No, I think that is what I meant. I was just so darned excited about
> knowing *why* this answer was true I mis-spoke. :-)
>
>> It makes me wonder though - have IDL pointers always behaved that way,
>> or is it a moving target (ala relaxed structure definitions post IDL 5.3
>> talked about in another thread) ?
>

> I think they have always behaved this way. But often these things are
> introduced without, uh, too much fanfare, and it takes everyone a while to
> catch up.

Doesn't surprise me... see the pointer tutorial on using this to pass
structure members by reference (sort of). In Ken's original example, it's
easy to just pass *data.array, and it will go in by reference.

JD

Subject: Re: Pass by value and performance
Posted by Antonio Santiago on Thu, 15 Dec 2005 06:56:50 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman wrote:
> Perhaps someone can clarify this for me.
>
> I was doing this
>
> data = {values : FLTARR(...), $
> other : other stuff ...}
>
> Then pass "data" to a procedure and do this
>
> result = INTERPOLATE(data.values, x, y, z)
>
> In this case data.values is passed by reference. (p. 92 of Building IDL
> Applications)
>

Be carefull (p. 93 of Building ...)

Note:
IDL structures behave in two distinct ways. Entire structures are
passed by reference, but individual structure fields are passed by
value. See Parameter Passing with Structures on page 189 for
additional details.

>
> If I were to use pointers, I could do something like this instead
>
> data = {values : PTR_NEW(FLTARR(...)), $
> other : other stuff ...}
>
> Pass "data" to a procedure and do this
>
> result = INTERPOLATE(*data.values, x, y, z)

>
> Is *data.values passed by reference or by value? And how does one tell?
>

I suposse yes. In the previous example:
```
  PRO JUNK, var
    var = var * 5
  END
```

And:

```
  IDL> a = Ptr_New(5)
  IDL> junk, *a
```

I like to understand pointers in IDL in this way:

1.- 'a' is a conventional variable managed by IDL and its "garbage collector".

2.- '*a' is a HEAP variable, where 'a' stores a reference to it. Also, the content of the variable 'a' is stored in the heap memory.

Then 'a' is a reference for a "normal" variable that stores a reference, and '*a' is a reference to a HEAP variable that stores a 5.

junk, *a --> The content of the HEAP memory variable is pased by value.

An then:

*a = *a * 5 is the same as b = b * 5

the diference is first is a HEAP variable and the second is a convential memory variable.

>
> Ken

Really, I have never question this until I saw your post. These are the good things of the net :)

--

Antonio Santiago Pï¿½rez
(email: santiago<<at>>grahi.upc.edu   )
(www: http://www.grahi.upc.edu/santiago)
(www: http://asantiago.blogsite.org   )

GRAHI - Grup de Recerca Aplicada en Hidrometeorologia
Universitat Politï¿½cnica de Catalunya
