Subject: Re: Code optimization
Posted by pgrigis on Wed, 16 Feb 2011 20:51:56 GMT
View Forum Message <> Reply to Message

On Feb 16, 3:38 pm, kisCA <ki...@hotmail.com> wrote:
> Hi there,
>
> I am trying, if it's possible to write thes lines in matrix formalism
> but it seems a bit tricky for me.
>
> FOR I=1,NUM DO BEGIN
> SUMI=0.
> FOR K=1,M1 DO BEGIN
> FOR J=1,M1 DO BEGIN
> SUMI=SUMI+COV(J-1,K-1)*CS(J-1,I-1)*CS(K-1,I-1)
> ENDFOR
> ENDFOR
> VAR_ALPHA(I-1)=SUMI*VAR_FACTOR(I-1)
> ENDFOR
>
> Is anyone got an idea it will be welcome
>
> Cheers

Yeah, it looks like that doesn't really need loops...

First you want to replace the inner loop with a matrix
multiplication of COV and CS - call the result R (make
sure you get the row and columns in the proper order).
Then you want to replace the K loop with total(R*CS,1).
Then you want to replace the I loop with a multiplication
of the last result by var_factor.

Something along those lines should work...

Ciao,
Paolo

Ciao,
Paolo

Subject: Re: Code optimization
Posted by Jeremy Bailin on Wed, 16 Feb 2011 20:57:54 GMT
View Forum Message <> Reply to Message

Are cov, cs and var_factor functions or variables? If variables, it would be a lot clearer to subscript them with [] instead of (). (also, why start those for loops at 1 and have -1 scattered throughout the code? it would be clearer to loop from 0 to num-1 or m-1 respectively and just index with i, j and k)

Assuming they're variables:

```
; calculate the matrix sum
sum = matrix_multiply(matrix_multiply(cov, cs), cs, /atranspose)
; pick out the diagonals, which are the ones with the same i in both cs multiplications
diags = (num+1) * lindgen(num)
; multiply by var_factor
var_alpha = sum[diags] * var_factor
```

-Jeremy.

---

Subject: Re: Code optimization
Posted by kisCA on Wed, 16 Feb 2011 21:23:17 GMT
View Forum Message <> Reply to Message

Thanks a lot, both of you. I will try your way.

Jeremy, I understand your disappointment about the subscript in the loops, but it's a code I get from a collegue who I suppose wrote it in Fortran and then just translate it in IDL. That's why I want to refresh it a bit to also better understand what's all about... And believe me, the rest was worst, it's the last part and I am exhausted ;-)

Cheers

---

Subject: Re: Code optimization
Posted by kisCA on Wed, 16 Feb 2011 21:25:39 GMT
View Forum Message <> Reply to Message

Jeremy, also, could you explain me a bit more the difference between [] and ().
cov, cs, var_factor are variables.
I am sorry I just moved to idl 3 month ago from matlab and I really have difficulty with "the subscript game" going on here...
Thanks

---

Subject: Re: Code optimization
Posted by Jeremy Bailin on Wed, 16 Feb 2011 21:48:35 GMT

View Forum Message <> Reply to Message

On Wednesday, February 16, 2011 4:25:39 PM UTC-5, kisCA wrote:
> Jeremy, I understand your disappointment about the subscript in the
> loops, but it's a code I get from a collegue who I suppose wrote it in
> Fortran and then just translate it in IDL.

Yeah, it definitely has that look to it. ;-)

> Jeremy, also, could you explain me a bit more the difference between
> [] and ().
> cov, cs, var_factor are variables.
> I am sorry I just moved to idl 3 month ago from matlab and I really
> have difficulty with "the subscript game" going on here...
> Thanks

No problem! A Long Time Ago, IDL used () for both function calls and array subscripts. But that's not a good idea, since it leads to ambiguity not just in reading code but also in executing it (what happens if you have both a variable and function with the same name?). So [] was introduced for array subscripts. Now you can use either by default, but [] is much preferred.

If you use compile_opt idl2, then you can *only* use [] (as well as a few other useful changes).

-Jeremy.

---

Subject: Re: Code optimization
Posted by Fabzou on Thu, 17 Feb 2011 09:14:12 GMT
View Forum Message <> Reply to Message

On 02/16/2011 10:25 PM, kisCA wrote:
> the difference between
> [] and ().

The use of () to access indexes in arrays is allowed in IDL for
backwards compatibility but should now only be reserved for functions.
In your code, you can prevent any error by writing

COMPILE_OPT IDL2

at the very beginning of each function/procedure you write. Accessing
variable arrays with () will not compile.

---

Subject: Re: Code optimization
Posted by kisCA on Thu, 17 Feb 2011 19:42:47 GMT
View Forum Message <> Reply to Message

Jeremy and Fabzou, thanks for these details I think I got it now, I
will read about compile_opt IDL2...

Jeremy, I am amazed how you manage so quickly to find a way to
optimize the code. Do you have some tips to achieve it?

Cheers

---

Subject: Re: Code optimization
Posted by Jeremy Bailin on Thu, 17 Feb 2011 19:52:28 GMT
View Forum Message <> Reply to Message

On Thursday, February 17, 2011 2:42:47 PM UTC-5, kisCA wrote:
> Jeremy, I am amazed how you manage so quickly to find a way to
> optimize the code. Do you have some tips to achieve it?

It obviously depends on the detailed situation, but the main thing is to understand what you're trying to do at a reasonably high level. So looking at the last statement in the outer loop and saying "that's a multiplication of two vectors", or looking at the inner 2 loops and saying "that's 2 matrix multiplications". Then it's often easy to see how that maps onto built-in IDL operators and functions.

-Jeremy.

---

Subject: Re: Code optimization
Posted by pgrigis on Thu, 17 Feb 2011 20:09:59 GMT
View Forum Message <> Reply to Message

On Feb 17, 2:42 pm, kisCA <ki...@hotmail.com> wrote:
> Jeremy and Fabzou, thanks for these details I think I got it now, I
> will read about compile_opt IDL2...
>
> Jeremy, I am amazed how you manage so quickly to find a way to
> optimize the code. Do you have some tips to achieve it?

Well technically Jeremy's solution is slightly suboptimal in that
it computes a matrix multiplication for 2 arrays, but only uses the
diagonal elements of the results.
If performance matters to you you may want to avoid doing that
2nd matrix multiplication and just compute the result directly
as I suggested.

Ciao,
Paolo

>
> Cheers

---

Subject: Re: Code optimization
Posted by Kenneth P. Bowman on Thu, 17 Feb 2011 20:52:12 GMT
View Forum Message <> Reply to Message

In article
<ff8c9322-c3c0-4ed0-b9c2-73ae36c71a16@g10g2000vbv.googlegroups.com>,
 kisCA <kis93@hotmail.com> wrote:

> Jeremy, I am amazed how you manage so quickly to find a way to
> optimize the code. Do you have some tips to achieve it?

Here is my short list of priorities for code optimization in IDL:

  Optimize where it matters - Don't worry about optimizing code
    that does not affect the overall cpu time.
  Do binary I/O - NetCDF, HDF, or binary, not ASCII.
  Access memory in order - Think about the order in which you are
    likely to access your data before you define arrays.  This will
    make better use of caches in modern cpus.
  Use IDL built-in functions and operators - There are many
    tricks for using REBIN, REFORM, and HISTOGRAM, but be aware
    that they are often memory intensive.
  Optimize your innermost loop - Loops are not necessarily bad
    in IDL if the calculations within a loop are well optimized.
  Don't do unnecessary calculations (e.g., move constants out of loops)
  Buy the IMSL library if you need those functions - Your time is
    probably much more expensive than an IMSL license.

If you keep those things in mind, you will rarely have to resort
to the code profiler or external modules.

This seems sure to start a discussion.  ;-)

Ken Bowman

---