Subject: Re: Quickest method for calculation
Posted by Karl[1] on Mon, 24 May 2010 15:58:00 GMT
View Forum Message <> Reply to Message

On May 24, 8:51 am, David Fanning <n...@dfanning.com> wrote:
> Gray writes:
>> However, instead of squaring, you should do y.rho*y.rho.
>
> Why is that?
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming:http://www.dfanning.com/
> Sepore ma de ni thue. ("Perhaps thos speakest truth.")

The IDL interpreter is probably going to compute y.rho^2 by calling the pow() function, which is a standard C library function. The pow() function probably uses exp(y * ln(x)) and probably needs to check/ handle bad input values for these functions. The pow() function itself may be optimized to spot the easy multiply or use some other faster calculation, but that is going to vary from platform to platform. In any case, is is easy to see that a single multiply is better at this point.

Also, there may be some float -> double and double -> float conversions going on, which are going to strain memory and CPU even further. The pow() function takes double arguments and returns double.

IDL has always been a "do what I say" language, and so it would probably expect you to write y.rho * y.rho instead of doing this optimization for you, which it could. But I'd guess that it doesn't in this case.

Minimizing memory access would be the other thing to look at. If you go with the multiply, you are going to read the y.rho array once to compute its square and then store the result in a temp. Then you will read the temp back again when multiplying it by the temp. You end up looping through some of the data twice. This is not the optimum cache behavior.

If it was Really Important for this to run fast, I would consider writing the function calculation in C so that I end up making only one pass through all the data:

```
for (i=0; i<n; i++)
  top_tem[i] += y.rho[i] * y.rho[i] * temperature[i]
```

There are a lot of docs and examples for coding functions in C that you can call from IDL, so this is not very hard to do and may be worth doing just to see how much faster it is.

If your compiler is really good, you may be able to turn on an option that generates SSE code, which may give you another 2x to 3x. The above pattern is an easy one for the compiler to recognize as SIMD-exploitable. If the compiler doesn't do this for you and you have the time, you can write the SSE code yourself.

But yeah, using the multiply instead of the ^ will have the most ROI.

Karl