
Subject: Re: Reducing an array.

Posted by [Craig Markwardt](#) on Wed, 02 Oct 2002 13:30:56 GMT

[View Forum Message](#) <> [Reply to Message](#)

"Dick Jackson" <dick@d-jackson.com> writes:

> "Craig Markwardt" <craigmnet@cow.physics.wisc.edu> wrote in message
> news:on65wnysmk.fsf@cow.physics.wisc.edu...

[...]

>> In this case you can use TOTAL() directly. First you REFORM() your
>> data into a 2-d array, 64xN, then then total the 1st dimension. This
>> works because each of your values has only one data bit set, so
>> summing and ORing are equivalent.

>>

>> compressed_data = reform(compressed_data, 64,
> n_elements(compressed_data)/64)
>> result = total(compressed_data, 1)
>>

>> That's it! For JD, I could have combined both statements onto one
>> line, but this is more readable.

>

> There's one problem with this, in that Total() returns a Double result
> at best (with the /Double keyword), but Joe wanted 64-bit integers. A
> 64-bit Double value with some bits used as exponent cannot represent as
> many distinct values as the 64-bit integer, so we will lose information
> here.

>

> Looks to me like this all has to be done in 64-bit integers. I'm sorry I
> can't find a *really* elegant solution for you right now, but if your
> data array is very large, then a single loop over 64 columns might not
> be too inefficient. Here's my best attempt (it does 100000 ints in 2.2
> seconds here, fast enough?):

Hi Dick--

You are right, TOTAL just won't preserve all the bits of precision.

I like your summation over the 64 bits. A loop with 64 iterations is
a FOR loop not even worth getting worried about. Good job!

By the way, I have found ISHFT to be faster than the exponentiation
operation, but in this case it only seemed to be about 20% faster.

Craig

--

Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
