Subject: Re: FOR loops removal
Posted by Chris[6] on Tue, 19 Aug 2008 20:04:04 GMT
View Forum Message <> Reply to Message

> Can anybody tell me why removing one loop doesn't help in this case or
> what i'm doing wrong?

I think the main reason why your second code snippet isn't much faster than the first is because it's not a very good vectorization (the term to describe the elimination of loops in favor of array based operations).

Not being a computer scientist, I don't actually 'understand' what IDL does when it compiles and runs code. But the image in my mind is akin this old man I saw in a post office one time. He couldn't really hear that well, and kept (loudly) asking the post office clerk 'when the hell those Rat-a-ville stamps are coming in' (after eavesdropping for a while, I realized that he was actually sent by his wife to buy stamps from the movie 'Ratatouille'). After the clerk (repeatedly) told him that a) it was pronounced 'rat-uh-too-eee' and b) they would get them next week, the old man was on his way. The impressive thing was that this 90 year old man FLEW out of the post office when he was done. He was fast - like Lolo Jones fast.

How does this connect? IDL for loops are slow because the part of IDL that interprets your file a fast but crotchety old man who can't hear you very well and may not even really be listening. Any time you tell him to do something, it takes him a while to interpret what you just said - much longer than other, less crotchety men. Once he figures out what's going on, however, he's plenty fast (especially if you tell him to do something that he was already designed to do, for which he has been well optimized). Good vectorization, then, minimizes the number of instructions (e.g. iterations in a loop) while maximizing the amount of work to do with each instruction.

Your second loop doesn't have any fewer iterations than the first loop - it just gets rid of one nested for loop and increases the size of the previous loop. Un-nesting the loops helps a bit (looping the loop is two layers of interpretation. IDL has no patience for such tasks. He lived through the depression and fought the Germans), but you really aren't following the principle of 'loop less with bigger processing chunks in each step.'

Wox's code is the right way to vectorize your loop. It truly iterates fewer times, and gives IDL more to chew with each line of instruction. I wouldn't bother eliminating the L loop. As soon as you do some hefty processing in each iteration, the looping penalty goes away, and you don't need to worry about your vectorization creating huge temporary

arrays and paying penalties in memory allocation.

As long as you don't have any loops where, at each iteration, you are simply accessing an element of an array, IDL should be pretty fast.

chris
