Subject: Re: Speeding up multiple file reading
Posted by JD Smith on Thu, 02 Feb 2006 16:56:52 GMT
View Forum Message <> Reply to Message

On Thu, 02 Feb 2006 08:44:51 -0700, David Fanning wrote:

> clivecook59@gmail.com writes:
>
>> I have a program where i need to read in multiple files. Currently i
>> read in 6000 binary files using a function i have written. This reads
>> three columns of data out of each file. To do this i use a loop that
>> calls the function to read a file who's data is then added to an array.
>> At the moment it takes around 90 seconds to go through the 6000 files.
>> Is there any way that i can read this data not using a loop? Or at
>> least are there any tips for speeding this up?
>
> What does your code look like for reading the three columns
> of data? This is likely to be the weakest link in your process.

When it comes to understanding why a given bit of IDL code is so slow, there really is no need to speculate, given how *easy* using the IDL profiling tool is. Just make sure your routines of interest are compiled (typically just by running the operation once), and type:

```
IDL> profiler,/system & profiler
IDL> run_my_code
IDL> profiler,/report
```

and you get instant access to a nice table summarizing the number of times each routine (user or system) is called, how long that took per call and in total, etc. Then work a bit on the slowest function(s), re-compile, type

```
IDL> profiler,/reset
```

to clear the last report, run the slow operation again, and get another report, checking whether your changes improved things.

Obviously, operations like a=[a,findgen(1000)] don't get a separate line item in the PROFILER report, but it's usually easy to figure out their impact by examining the "Only" column of the report, which tallies the time spent only in the body of a routine, not in routines it calls. So if you have a routine like

```
pro my_routine
  a=do_something(FOO=foo)
  b=readu(a,foo)
  ...
  for i=0,n_elements(foo)-1 do b=[b,randomu(sd,1000)]
end
```

it's fairly easy to figure out what that repetitive array concatenation is doing to you by looking at the "Only" column for MY_ROUTINE.  If there is too much "body" in a routine to easily tell where the problem spots are using this method, consider breaking the body up into some temporary routines which PROFILER can track.

I've just begun using the PROFILER regularly, and really appreciate how useful it is.  For instance, if you have a complex GUI operation which is slow, running in a widget in the background, just invoke profiler, run the slow operation, generate the report, find and fix the slow parts, recompile, and repeat -- all without ever quitting the running widget program!  Try that in other languages.

Note that unfortunately the docs for PROFILER are a bit difficult to parse.  It took me forever to understand that:

```
IDL> profiler
```

adds all compiled user routines to the list of things to profile, while:

```
IDL> profiler,/SYSTEM
```

adds only system routines, and nothing else.  So, if you want *both* system routines and user routines to be profiled, you need two calls to PROFILER.  This is also true of clearing the profiling, i.e., to totally shut down all profiling, you need:

```
IDL> profiler,/CLEAR,/SYSTEM & profiler,/CLEAR,/RESET
```

to do the job properly.  That is, "profiler,/SYSTEM" and "profiler" function like two completely separate tools, except for /RESET, which resets everything.  Not exactly intuitive, but I can forgive them that for providing us such a quasi-miraculous profiling tool.

JD