Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Wed, 10 Jul 2013 19:24:49 GMT
View Forum Message <> Reply to Message

Alright, so I tried looking at a couple things with the following code:

```
pro memissue
  mbegin=(memory())[0]
   ; create the object outside the loop (no difference memory-wise...just faster to run)
  o = obj_new('IDL_IDLBridge',output='out_memissue.asc')
   ; write bridge memory
  o->execute, 'print, "ini", (memory())[0]'
  for i=0,100 do begin
     m0 = (memory())[0]
     o->Execute, 'x = 1'        , /nowait

      ;wait for execution to complete (I doubt this is needed for 'x=1'...but just incase)
     while (o->status() ne 0) do wait, 0.1
     ;Nata's additions----
     all_var='tmp_var'
     o->Execute, all_var+'=ROUTINE_INFO("$MAIN$",/VARIABLES)'
     all_var=o->GetVar(all_var)

     command='DELVAR, '+STRJOIN(all_var,', ')
     o->Execute, command
     ;End of Nata's additions-----

     m1 = (memory())[0]
     if (i gt 0) then begin
        print
        print,'iteration: ',i
        print,'Memory lost in this step thru loop: ', m1 - m0
        print,'Total memory lost since beginning:  ',m1-mbegin
        print
     endif
      ; write iteration memory
     o->execute, 'print, '+strtrim(i,2)+', (memory())[0]'
  endfor
   ; write final memory
  o->execute, 'print, "fin", (memory())[0]'
   ;wait for any remaining execution to complete (really shouldn't be needed)
  while (o->status() ne 0) do wait, 0.5
   ; destroy object
  obj_destroy, o
;  o->Cleanup
  mfinal=(memory())[0]
  print, 'Total run memory difference:', mfinal-mbegin
```

end

From what I've seen, nothing is gained by deleting the variables within the bridge (the bridge output file has 768987 for (memory())[0] at each iteration regardless of variable being deleted or not, which doesn't seem quite right to me). As far as the difference in memory between start and end of the parent: 210245 regardless of if I include "o->Execute, command" or not. If I comment out "/nowait" I find that every iteration has a 0 for memory lost and the total run difference is 2421. Without "/nowait" the bridge memory returns 769163 every iteration. So to my untrained eye, it seems like execute with "/nowait" produces something (maybe some sort of placeholder?) that isn't taken care of before moving on, but I have to admit that's just speculation on my part. On the upside, it seems to be restricted to only execute calls that are asynchronous (at least on my machine).
