

Re: Python threading

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2007-03/msg01093.html>

- *From:* "Gabriel Genellina" <gagsl-py2@xxxxxxxxxxxxx>
 - *Date:* Fri, 09 Mar 2007 02:56:53 -0300
-

En Fri, 09 Mar 2007 00:38:55 -0300, <test.07@xxxxxxxxxx> escribió:

pythoncom.CoUninitialize(), if I am not mistaken, releases a COM object (and therefore the memory it uses I assume).

Not exactly. *You* must release the COM object (usually assigning None to all references to it). CoUninitialize releases all system resources held by the COM application, unloads DLLs, and so; it *may* release all COM objects but I'm not sure.

When the command is run (in PythonWin for example), the memory is not released but persists for quite a while. Using my watch and the Task Manager in Windows I can see that the memory is released approximately 30 seconds AFTER I run the pythoncom.CoUninitialize() command is run.

If it is eventually released, you don't have a problem, right?

And, if it will be needed again after a while (when you run the next command), it looks better this way. There are many layers on memory management: the OS manages 4K pages that are mapped on the process address space; the C runtime suballocates such pages as memory blocks; Python itself has two levels of memory allocation (and I don't know the details). It's not easy to know exactly how much memory is actually used; I think the Python runtime, once it allocates a memory block, never releases it to the OS (remaining as a free block, available for further usage).

And in general, DLLs are *not* unloaded as soon as its usage count goes to 0; there is certain delay. (I don't remember, but I think there is a registry key called DLLUnloadTime or something). The idea is to avoid removing it from memory just to load it again a few moments later, so the unloading is delayed for a few seconds. You may be seeing this effect.

What memory do you want to release "right away"?

The memory I want to release is the memory the COM object used (the one initialized with the win32com Dispatch command). The "right-away" is not much of an issue, but if I can release it before I run each

Re: Python threading

command from the COM object that leaks memory, it would be nice.
Running upwards to 800MBs of RAM for one 500 line python script seems
a little bit too much for me.

Let's see if I can understand the problem. You run multiple commands on a COM object, one after another. The COM object is buggy and leaks some memory. You run each command on a separate thread, with a CoInitialize(), waiting for completion, and a CoUninitialize() at the end. Memory usage grows to 800MB when you invoke each command, but if you wait enough time after the thread finalizes, memory usage goes down to normal. After that, you run the next command, and so on.
Or do you wait until the whole program finalizes, and 30 seconds after that, memory usage drops?

—

Gabriel Genellina

.