VisualAPL in a 64-Environment

As a .Net language VisualAPL requires no proprietary memory management scheme because the .Net CLR (common language runtime virtual machine) manages virtual memory automatically. It is still interesting to examine how memory is allocated to a VisualAPL application, especially one that creates large objects.

Several measures of memory usage are available in .Net. In this example the 'PrivateMemorySize64' property of the current process is used to determine available memory for a process. This property is reported as an Int64 datatype.

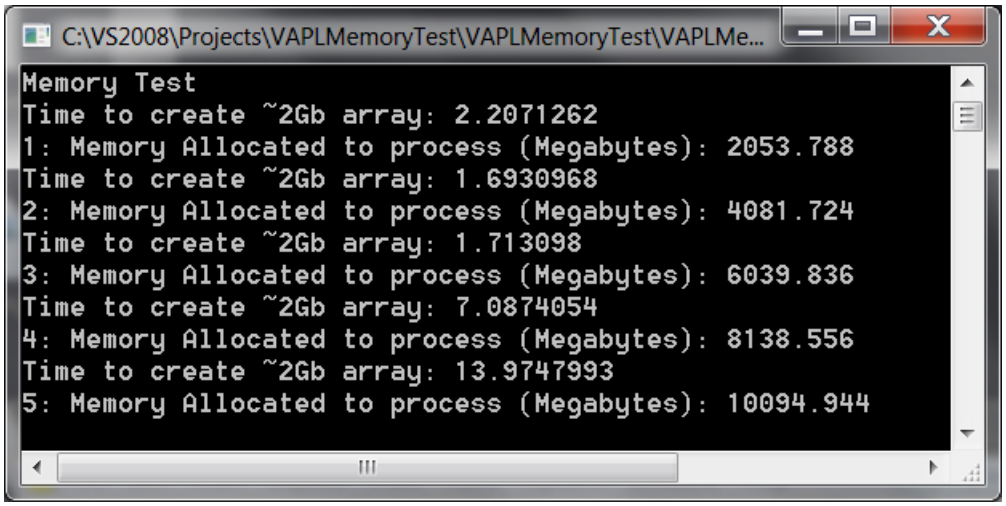
In this example large APL arrays are created consecutively and each time such an array is created, the memory allocated to the current process is displayed. In addition the approximate time necessary to create each of these arrays is also displayed. The unit of this time interval are not important for this example, so the difference in □AI[2] before and after an array is created is used.

Since .Net is managing the virtual memory allocated to the current process, when available physical memory is exhausted, .Net begins to utilize hard disk space as additional virtual memory. The time interval information results in this example make it easy to see when this happens.

This time interval information also makes it easy to see that in .Net with 64-bit hardware and operating system software, the total memory allocated to a process is not limited to the 2-Gb of a 32-bit system. This means that in the 64-environment and with sufficient memory installed, it is possible to have multiple objects each utilizing up to 2Gb in a VisualAPL application.

The illustrated results were obtained using a 64-bit workstation using Windows7 64-bit operating system and 8Gb of physical memory installed. Approximately 7.68Gb of physical memory is available in excess of that consumed by the operating system. Each object created in this example is an APL numeric vector of doubles with 250 million elements. Each such object consumes approximately 1.96Gb of memory.

After approximately 6Gb of physical memory are allocated to the first three APL arrays, the time to create the fourth APL array increases dramatically indicating that the available physical memory has been allocated and hard-disk-based virtual memory is now being allocated. The first three APL arrays can fit in the 7.68Gb of physical memory available to an application system, but the fourth and fifth cannot. .Net will continue to allocate hard-disk-based virtual memory until the user-defined limit on such memory would be exceeded.



For this example a VisualAPL ‘Console Project’ was used. The APL ‘execute’ operator creates the large APL arrays and the Console.WriteLine() method is used to display the time and allocated memory values: