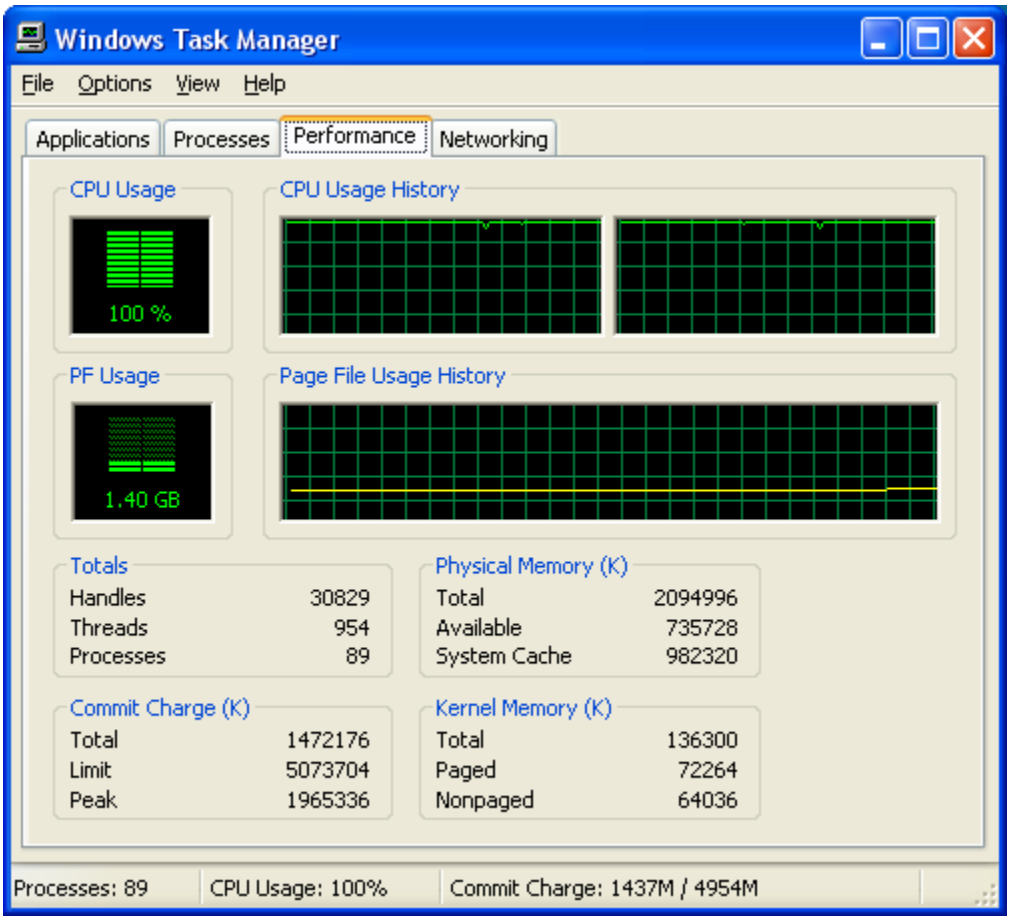
Parallel Geoprocessing in ArcGIS Desktop 9.3.1

Many people ask how ArcGIS Desktop can leverage their investment in hardware so that the available processors are all used. For example, how do you make use of a 4-core processor in a Desktop machine? This document and associated materials provides an example of how to make a multi-core Desktop machine into a parallel geoprocessor. The Python-based script tool included in this download is generic and may be modified to parallelize any suitable task. While a dual-core laptop (used to develop this example) is never going to be a powerful geoprocessor, this is the experience you can achieve by parallelizing geoprocessing:



ArcGIS Desktop 9.3.1 applications are single threaded; one process runs to support the user interface and geoprocessing for each application instance that is open. This means that only one CPU core is in use by ArcGIS at any moment. In the default situation where an ArcGIS process has a shared affinity with all CPU cores, then you will see an averaged single CPU equivalent usage across all cores at most, and no more.

Not all geoprocessing tasks are feasible to parallelize. This approach to parallelization will be to split an input into multiple parts to be run concurrently in separate processes; the geoprocessing task you wish

to parallelize must be supportable this way. This is called an “embarrassingly parallel” problem in relevant literature. To meet this requirement it must not matter which process handles any input feature or table row. The supplied example demonstrates geocoding; other examples would be clipping, buffer without dissolve, and point in polygon overlay. You must determine if your geoprocessing task may be parallelized. Note that the overhead in splitting and recombining the data being processed may mean you achieve little or no benefit when working with modest amounts of data and few processes.

ArcGIS Desktop applications are all subject to the memory architecture of 32-bit Windows; 4GB of physical memory is addressable, paging permits any process to make use of 2GB of this, with another 2GB of logical addresses in page files. A total of 4GB of page file is available to all processes. Once memory is exhausted a process requesting more will fail. ESRI recommends that users who want a truly scalable geoprocessing environment use ArcGIS Server, which can manage multiple computers and avoid these issues.

Now we have the “gotchas” out of the way, we can go into details. The supplied Python script **ParallelGeocodingExample.py** has been written to be generic, usable in a script tool that accepts a single table or feature class as input and outputs a derived feature class. The actual geoprocessing job that the script tool does is embedded in the script; you can edit the script to replace the geocoding functionality with what you want. Child processes are Python programs running scripts dynamically written by the script tool. Look for the comment line: “Begin code block for GP process of interest:” to make your edits. Be careful to handle the quoting and newline elements of strings, as they must be valid statements when executed later. Apart from editing to accomplish the geoprocessing task you want, you will need to edit the script to set the variable “wkPath” to a suitable value for your system and the paths referencing the locator to agree with where you download to. Now you may parallel geoprocess.