How do I keep thread pool threads, or other threads in general, from competing with my render thread for CPU?

devblogs.microsoft.com/oldnewthing/20170309-00

March 9, 2017

Raymond Chen

Processor affinity lets you specify which processor or processors a thread can run on. This works for threads you control, but what about threads you don't control?

Consider a game. You want your render thread to run with as little interference as possible because it is time-critical. You can set the processor affinity for the render thread to one of the processors, and set the processor affinity for all the other threads you create to other processors. That way, none of your other threads will compete for CPU resources with the render thread.

But what about threads that aren't yours? Suppose the thread pool creates a thread. Well, you don't get a chance to set that thread's affinity. It defaults to the process affinity, which means that depending on how lucky or unlucky you are, those threads might end up running on the same CPU as your precious render thread.

Enter CPU sets.

What you can do is call `GetSystemCpuSetInformation` to identify the available CPU sets. Assign your render thread to one of the CPU sets, and set the process default CPU sets to all the other CPU sets. The process default CPU sets are used by a thread which has not been explicitly assigned any CPU sets. The thread pool threads, therefore, will run on the process default CPU sets, which means that they won't try to run on the same CPU as your render thread, because you carefully set things up so that the render thread runs on a dedicated CPU set.

Raymond Chen

Follow