**Threading in .NET**

**Why thread is needed?**

First let us understand what was scenario in earlier time. In earlier time, there was only one thread, which was handling both operating system and application. As there was only one thread, long running task would prevent other task from execution and some task may contain bug, which will result in infinite loop and termination of machine where user have to restart computer which will result in loss of any data processed by any application running on computer.

**What is thread?**

Job of thread is to virtualize the CPU. Thread allow you to get maximum utilization of CPU. Windows give each process its very own thread. Process is instance of application that is using collection of resources. For example, if you open new word document then new document is process of word application.

**Advantage of thread**

**Performance**

If you are using multiple CPU for one application then you can schedule one thread per CPU which will allow to perform multiple operation of single application at single time. Window gives each process its very own thread that allow windows to be responsive even if long-running task is in execution.

**Multiple application(Responsiveness)**

We can easily access multiple application as windows gives each process its very own thread so if one application enters to infinite loop it will not prevent user from using other application and user can also kill frozen process with help of task manager.

**Disadvantage of thread**

**Memory**

Threads have time and space overhead associated with them!! Thread is used for local variables and argument passed to methods. In addition it also contain address indicating what thread should execute next when current thread returns. By default, windows allocate 1MB of memory for this kind of thread.

**Unmanaged DLL**

Windows has policy that all unmanaged DLL must be called whenever thread is created/destroyed for specific process. Unmanaged DLL are called for special initialization and clean up. Now consider situation where process have 200 unmanaged DLL, in this scenario 200 unmanaged DLL are called at start of thread and 200 unmanaged DLL are called at destroy of DLL.

**Context Switching**

Another problem with thread is context switching. One thread execute for specific time-slice, once time-slice is over, CPU store current value of thread, chose one thread from set of existing threads and load value of that thread into CPU. Context switching is pure overhead; that is, there is no performance or memory benefit form context switching.

**Solution:**

At the end of time-slice, CPU can allow to execute same thread, which will improve performance significantly.

Thread can voluntary ends earlier, which happens quite frequently. Consider one program in .NET, which have methods with parameter, and one thread is associated with this method. Now this thread will be in wait mode until method is called and once task is completed, it will again go in wait mode until method is called again.

Happy learning and wish you had a great day!!!