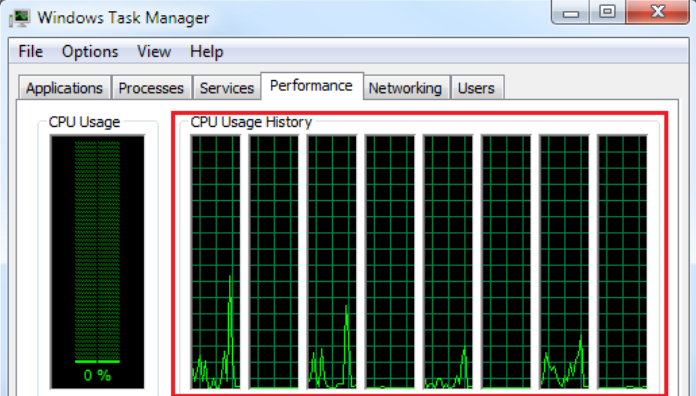
**Part 97 - Performance of a multithreaded program**

In this video we will discuss the performance implications of a multithreaded program when run on a single core/processor machine versus multi core/processor machine.

**First let's discuss, how to find out how many processors you have on your machine?**  
There are several ways  
**1.** Right click on the Task bar and select "Start Task Manager" option from the context menu. Click on the the performance tab. The number of green boxes under "CPU Usage History" should correspond to the number of processors on the machine.



**2.** The following code can be used in any .net application to find out the total processors on the machine  
Environment.ProcessorCount

**3.** On the windows command prompt window, type the following  
**echo %NUMBER\_OF\_PROCESSORS%**  
  
**On a machine that has multiple processors**, multiple threads can execute application code in parallel on different cores. For example, if there are two threads and two cores, then each thread would run on an individual core.This means, performance is better.  
  
If two threads take 10 milli-seconds each to complete, then on a machine with 2 processors, the total time taken is 10 milli-seconds.

**On a machine that has a single processor**, multiple threads execute, one after the other or wait until one thread finishes. It is not possible for a single processor system to execute multiple threads in parallel. Since the operating system switches between the threads so fast, it just gives us the illusion that the threads run in parallel. On a single core/processor machine multiple threads can affect performance negatively as there is overhead involved with context-switching.  
  
If two threads take 10 milli-seconds each to complete, then on a machine with 1 processor, the total time taken is 20 milli-seconds + (Thread context switching time, if any)

**Program used in the demo:**

usingSystem**;**

usingSystem.Diagnostics**;**

usingSystem.Threading**;**

classProgram

**{**

publicstaticvoidMain**()**

**{**

Stopwatchstopwatch=Stopwatch.StartNew**();**

ThreadT1=newThread**(**EvenNumbersSum**);**

ThreadT2=newThread**(**OddNumbersSum**);**

T1.Start**();**

T2.Start**();**

T1.Join**();**

T2.Join**();**

stopwatch.Stop**();**

Console.WriteLine**(**"Total milliseconds with multiple threads = "

+stopwatch.ElapsedMilliseconds**);**

Console.WriteLine**();**

stopwatch=Stopwatch.StartNew**();**

EvenNumbersSum**();**

OddNumbersSum**();**

stopwatch.Stop**();**

Console.WriteLine**(**"Total milliseconds without multiple threads  = "

+stopwatch.ElapsedMilliseconds**);**

**}**

publicstaticvoidEvenNumbersSum**()**

**{**

doublesum=0**;**

for**(**inti=0**;**i<=50000000**;**i++**)**

**{**

if**(**i%2==0**)**

**{**

sum=sum+i**;**

**}**

**}**

Console.WriteLine**(**"Sum of even numbers = {0}"**,**sum**);**

**}**

publicstaticvoidOddNumbersSum**()**

**{**

doublesum=0**;**

for**(**inti=0**;**i<=50000000**;**i++**)**

**{**

if**(**i%2==1**)**

**{**

sum=sum+i**;**

**}**

**}**

Console.WriteLine**(**"Sum of odd numbers = {0}"**,**sum**);**

**}**

**}**