How to decide pool size for Thread Pools

Thread pool is really an important topic to understand, if you are developing an application, which is designed with multithreading approaches. Thread pool is a pool of live worker threads. Worker thread is thread which accept a task, complete it and come back to thread pool to accept another task. Such way application can use already exist threads multiple time instead of creating new thread every time. By using thread pool in your application, you can boost its performance and utilize resource efficiently.

But in this article, we are not talking how to create thread pool and how thread pool works. For this you can read my other blog about thread pool. Right now, we are just discussing about, how to decide pool size for thread pool? I mean, on which factors we could decide that how many threads, we should create in our thread pool for better time and resource utilization.

Most of the time people don’t think about pool size for their thread pool and they create any number of threads in pool with their choice without thinking much on it. But believe me, if you don’t know how to decide number of threads in pool that could be more dangerous and could spoil your performance and memory management badly. Therefore, we will discuss in this article on which factors thread pool size depends and how an appropriate pool size affects performance of your application?

**There are few factors mentioned below on which thread pool size depends:**

**1. Available Processors:**This is first and most important factor on which thread pool size matters a lot. Because the purpose behind creating thread pool or making any program multithreaded is to utilize all available resources like processors and unused CPU cycles. If your pool size would be less than available processors in your system means you are not using all available processors and not utilizing resources fully. On other side, if your pool size is greater than your available processors that means you are creating more threads which a thread pool can handle. It is not possible to provide CPU cycle to each thread that means you are wasting memory by creating such threads in pool which is not going to utilize by thread pool. There is one more overhead for thread pool to maintain scheduling for such extra threads. Therefore, ideal pool size is available processors (AP) in your system or AP+1. Here is an example, how to get number of processors available in your system using Java.

**int poolSize  = Runtime.getRuntime().availableProcessors();**

OR

**int poolSize  = Runtime.getRuntime().availableProcessors() + 1;**

This is ideal pool size, if your multithreaded task is kind of computation, where threads are not getting block, wait on I/O or some combination.

But if your task also includes some kind of blocking or waiting time then there is one more way to determine pool size for such tasks. That is explained in next section.

**2.Behaviour of Tasks:**It is important to understand behaviour of your task what you want to perform inside thread. If you have different category of tasks with different behaviours that consider you thread pool size according to that behaviour. If your task is simple computation and do not obtain any blocking or waiting than you can consider number of available processors (AP)  or AP+1, as mentioned in above section.

But for tasks that also include I/O or other blocking operations, you need a larger pool size for that, since not all of the threads will be scheduled at all times, some will be in wait condition. In order to size the pool properly, you must estimate the ratio of waiting time to compute time for your tasks, this estimate need not be precise and can be obtained through profiling or instrumentation. Alternatively, the size of the thread pool can be tuned by running the application using several different pool sizes under a benchmark load and observing the level of CPU utilization.

The optimal pool size for keeping the processors at the desired utilization of CUP is:

**N = Number of processor available in system.**

**U = Target utilization of CUP; 0 >= U <= 1.**

**W/C = Ration of wait and computation time.**

Number of threads for thread pool can evaluate by this formula:

**Number of Threads = N \* U \* (1 + W/C)**

This is way you can estimate your suitable thread pool size.

**3.Amdahl’s Law:** In application development there are lots of tasks which cannot be perform totally concurrently, there are few task which need to be perfume sequentially. Therefore it is important to understand, how much proportion of task can be executed concurrent and how much speed you would get after making that portion of task concurrent. Therefore, Amdahl’s law is very useful to determine how speed up you would get if you are breaking up your task into parallelism and sequential.

According to Amdahl’s Law, if **P** is the proportion of task can be executed parallel then maximum speed up can get with **N**number of processors (threads) is:

**Speed up = 1/ (1-P) + P/N**

By applying Amdahl’s law you can determine how many threads can give you better result in case of concurrency.