



# Systemy rozproszone

Informatyka, sem. 6  
część II

## Thread Pools

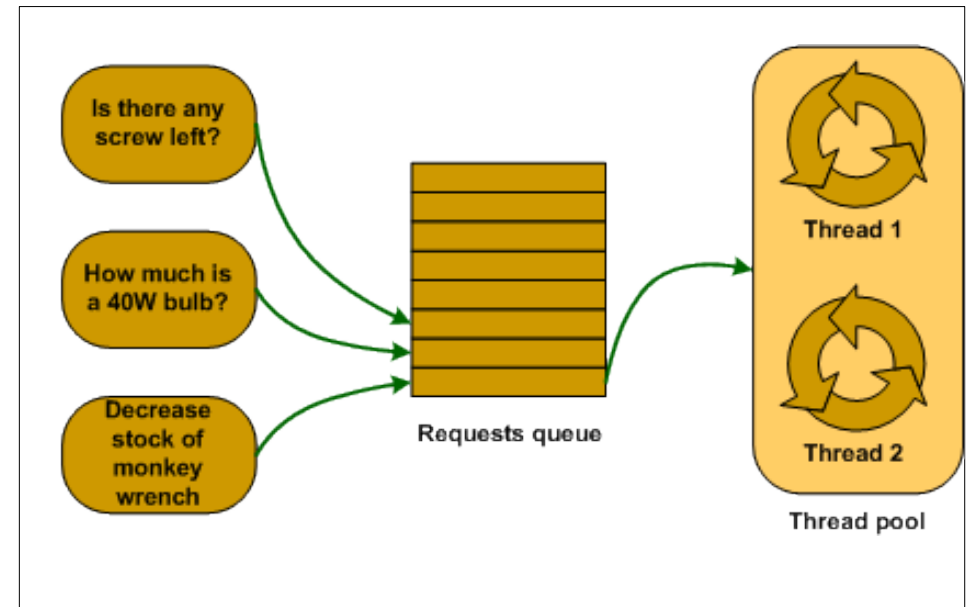
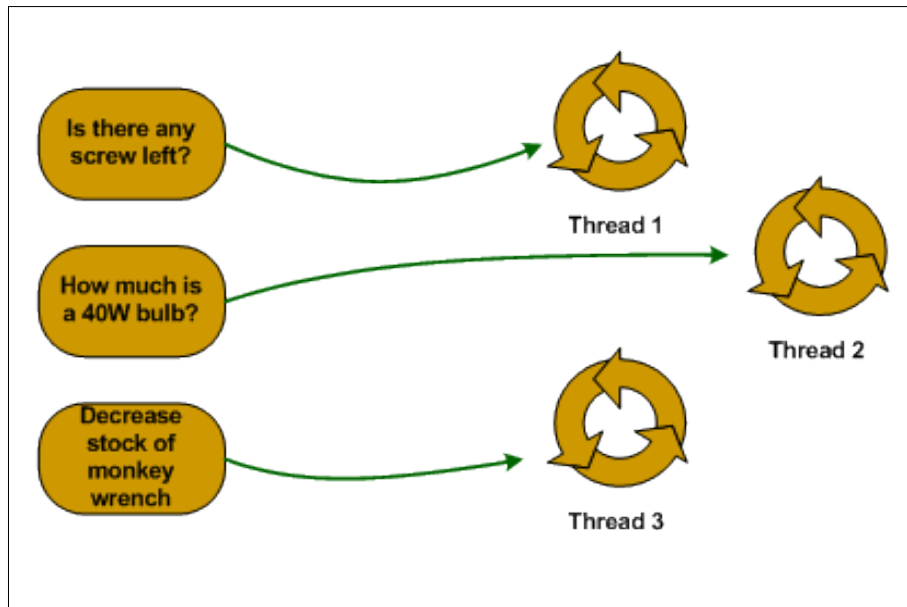


# Wielowątkowość a pula wątków

- Uruchomienie oddzielnego wątku dla każdej żądanej operacji
  - narzut na przełączanie wątków
  - wykładniczy wzrost czasu odpowiedzi od liczby żądań, większy czas pracy CPU w trybie uprzywilejowanym (przełączanie wątków)
- Pula wątków
  - nadchodzące żądania są ustawiane w kolejkę i przekazywane do działających wątków
  - liczba działających wątków może być optymalizowana
    - np. wątki czekają na operacje we/wy -> małe zużycie procesora -> uruchomienie kolejnych wątków



# Wątki a pula wątków



- Wątki

© 2009 Microsoft Corporation

## Pula wątków



# Thread Pool w środowisku .NET

- .NET framework udostępnia mechanizm Thread pool
  - pula wątków tworzona przy pierwszym dostępie
  - jedna pula dostępna dla wszystkich aplikacji (application domains) w ramach jednego procesu
  - klasa ThreadPool, namespace System.Threading
  - ThreadPool.QueueUserWorkItem (WaitCallback callBack, object state) - uruchomienie funkcji w puli
  - public delegate void WaitCallback (object state) - parametr dla funkcji uruchamianej w puli



# Thread Pool demo



- static void PooledFunc(object state)
- Thread.CurrentThread.IsThreadPoolThread, GetHashCode());
- Sleep vs busy wait
- Monitorowanie ThreadPool
  - GetAvailableThreads
  - GetMaxThreads



# Timer execution

- Wywołanie zadania w określonym czasie
  - wykorzystanie WM\_TIMER i aplikacji okienkowych, ale przetwarzanie wiadomości jest mało dokładne
- Zaimplementowanie "ręczne" przez wątek
  - narzut przy większej liczbie wątków
- Wysłanie żądania do ThreadPool
  - time-dependant zamiast aktywnego czekania
  - System.Threading.Timer - umożliwia określenie delegate dla periodycznego wywołania



# Timer demo

- public Timer(TimerCallback callback, object state, int dueTime, int period);
  - state - parametr dla funkcji, dueTime - opóźnienie do uruchomienia, period - czas między wykonaniami



# Asynchroniczne wywołanie operacji we/wy



- .NET umożliwia wywołanie operacji we/wy asynchronicznie
  - nie wymagają CPU, mogą trwać dłuższy czas
  - parametr `useAsync` podczas tworzenia strumienia pliku (`FileStream`)
- Dla plików `BeginWrite`, `IAAsyncResult` interface, `AsyncWaitHandle`
- Analogiczne operacje dla socketów
  - `BeginReceive`, `BeginSend`, `BeginConnect`, `BeginAccept`