Another more efficient solution to the problem of a long-running task running on the thread pool persistent thread

devblogs.microsoft.com/oldnewthing/20170220-00

February 20, 2017

Raymond Chen

Last time, we broke up a series of operations on the persistent thread pool thread so that the thread pool did the waiting rather then clogging up the persistent thread with a synchronous wait. This does still have the downside of processing the notification on the persistent thread, which could be a problem if that processing is time-consuming.

What we can do is to use the persistent thread pool thread only for things that absolutely must be done on a persistent thread, and move everything else to a thread pool task thread.

Make the highlighted changes to the code we had from last time.

```
// Error checking elided for expository purposes.

void WidgetMonitor::RegisterNotificationWait(
 void* parameter)
{
 WidgetNotificationContext* context =
   reinterpret_cast<WidgetNotificationContext*>(parameter);
 RegisterWaitForSingleObject(&context->waitHandle,
  context->registryEvent,
  WidgetNotificationWaitCallback,
  context,
  INFINITE,
  WT_EXECUTEONLYONCE ~~/* | WT_EXECUTEINPERSISTENTTHREAD */~~);
 RegNotifyChangeKeyValue(context->hkey, false,
                         REG_NOTIFY_CHANGE_LAST_SET,
                         context->registryEvent, TRUE);
}

void WidgetMonitor::WidgetNotificationWaitCallback(
    void* parameter, BOOLEAN /* TimerOrWaitFired */)
{
 WidgetNotificationContext* context =
   reinterpret_cast<WidgetNotificationContext*>(parameter);

 ... process the change ...

 QueueUserWorkItem(RegisterNotificationWait,
                   context,
                   WT_EXECUTEINPERSISTENTTHREAD);
}

void WidgetMonitor::StartMonitoring()
{
 auto context = new WidgetNotificationContext();
 context->hkey = ...;
 context->registryEvent = ...;
 QueueUserWorkItem(RegisterNotificationWait,
                   context,
                   WT_EXECUTEINPERSISTENTTHREAD);
}

void WidgetMonitor::StopMonitoring(
    WidgetNotificationContext* context)
{
 // WARNING! Massive race conditions here need to be addressed.

 if (context->waitHandle) {
  UnregisterWait(context->waitHandle);
  context->waitHandle = nullptr;
 }
 ... clean up other resources ...
 delete context;
}
```

What we did this time was to put only the `RegNotifyChangekeyValue` on the persistent thread. Everything else runs on a normal thread pool thread. That way, we minimize the amount of code running on the persistent thread.

The last fix we can make is to take advantage of a new feature in Windows 8: The `REG_NOTIFY_THREAD_AGNOSTIC` flag, which turns off the old behavior of stopping the notification when the thread exits. With that change, we don't need the `WT_EXECUTEINPERSISTENTTHREAD` flag at all.

```
// Error checking elided for expository purposes.

void WidgetMonitor::RegisterNotificationWait(
 WidgetNotificationContext* context)
{
 RegisterWaitForSingleObject(&context->waitHandle,
  context->registryEvent,
  WidgetNotificationWaitCallback,
  context,
  INFINITE,
  WT_EXECUTEONLYONCE ~~/* | WT_EXECUTEINPERSISTENTTHREAD */~~);
 RegNotifyChangeKeyValue(context->hkey, false,
                         REG_NOTIFY_CHANGE_LAST_SET |
                         REG_NOTIFY_THREAD_AGNOSTIC,
                         context->registryEvent, TRUE);
}

void WidgetMonitor::WidgetNotificationWaitCallback(
    void* parameter, BOOLEAN /* TimerOrWaitFired */)
{
 WidgetNotificationContext* context =
   reinterpret_cast<WidgetNotificationContext*>(parameter);

 ... process the change ...

 RegisterNotificationWait(context);
}

void WidgetMonitor::StartMonitoring()
{
 auto context = new WidgetNotificationContext();
 context->hkey = ...;
 context->registryEvent = ...;
 RegisterNotificationWait(context);
}

void WidgetMonitor::StopMonitoring(
    WidgetNotificationContext* context)
{
 // WARNING! Massive race conditions here need to be addressed.

 if (context->waitHandle) {
  UnregisterWait(context->waitHandle);
  context->waitHandle = nullptr;
 }
 ... clean up other resources ...
 delete context;
}
```

Raymond Chen

Follow