

Re: Win32 Thread Interleaving

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2007-01/msg00308.htm>

- *From:* "Pavel Lebedinsky [MSFT]" <pavel@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 23 Jan 2007 03:58:48 -0800
-

The reason you're seeing different behavior on WS03 SP1 is probably because of the changes to critical sections that were implemented in that release:

<http://www.bluebytesoftware.com/blog/PermaLink.guid.e40c2675-43a3-410f-8f85-616ef7b031aa.aspx>

I agree with PLS – the way this code worked in XP (with two threads moving in lockstep) was not very efficient to begin with. Generally, designs where threads synchronize with each other only infrequently (say, to obtain the next work item) and most of the actual work is performed work outside of critical sections are going to perform much better (on any OS).

--

This posting is provided "AS IS" with no warranties, and confers no rights.

"PLS" wrote:

It's a design problem. You should not have both threads running all the time. Not only does that make scheduling difficult for your program, it disrupts all other programs running on the machine.

ThreadA should wait on a semaphore until there is work for it to do. This is key, this thread should be waiting unless it's busy processing a unit of work.

ThreadB should wait while listening for the broadcast, then queue the work and signal the semaphore. Again, the thread should not be running all the time.

I read some post for win32 thread application and thought if you guys can hep us.

we are developing trading platform for different exchanges and facing a strange problem. Our application creates two threads both threads are

Re: Win32 Thread Interleaving

running
for indefinite times (means there is no sleep in thread loop function).
Thread 1 will always check the queue for possible work to be done and
Thread
2 always listens the network broadcast.

The code is something similar to

```
void *ThreadAFunct(void *arg){
While(true){

int workAvailable = CheckQueue();
//....
if(workAvailable){
LockMutex(sharedMutex);
//do some work
UnLockMutex(sharedMutex);
}
//.....
}
}

void *ThreadBFunc(void *arg){
while(true){
//.....
LockMutex(sharedMutex);
//ListenForBroadcast...
UnLockMutex(sharedMutex);
//.....
}
}
```

Both threads uses same mutex. and for acquiring and releasing mutex we are using EnterCriticalSection and LeaveCriticalSection functions.

Now problem we are facing is, in windows XP hyper threaded processor, threads get the mutexes in just a few milliseconds and never had any problem with thread interleaving, threads releases after few milliseconds and other thread acquires mutex in very little time. But we are seeing different behaviour in windows 2003 server (SP1) hyper threaded machine. Thread doesn't interleave perfectly like XP, sometimes a thread runs for long time compare to other thread (I used timers and found out that Thread B gets most of the processor time compare to thread A.)

Re: Win32 Thread Interleaving