Subject: MT anomaly...
Posted by mirek on Wed, 16 Apr 2008 10:29:09 GMT
View Forum Message <> Reply to Message

I am now working on some advanced "MT topics" and have encountered this anomaly:

```
#include <Core/Core.h>

using namespace Upp;

#ifdef PLATFORM_POSIX
__thread int threadid;
#else
__declspec(thread) int threadid;
#endif

#define LLOG(x)  LOG((threadid) << " " << x << ", count " << count)

RWMutex          rwlock;
VectorMap<int, String> cache;

String Fn(int x)
{
 return AsString(sin(sqrt((double)x)));
}

void CheckResult(int x, const String& r)
{
 if(r != Fn(x)) {
  DUMP(r);
  DUMP(Fn(x));
  Panic("Failure! " + AsString(threadid));
 }
}

int writes, removes;

void WorkThread(int id)
{
 threadid = id;
 for(int i = 0; i < 200000000; i++) {
  if(i % 10000 == 0)
   INTERLOCKED
    Cout() << id << ": " << i << ", writes: " << writes << ", removes: " << removes << "\n";
  int x = rand() & 0x7fff;
  rwlock.EnterRead();
  int q = cache.Find(x);
```

```
  if(q >= 0) {
   String r = cache[q];
   CheckResult(x, r);
   for(int i = 0; i < 100; i++)
    Fn(x);
   rwlock.LeaveRead();
  }
  else {
   rwlock.LeaveRead();
   rwlock.EnterWrite();
   q = cache.Find(x);
   if(q >= 0)
    CheckResult(x, cache[q]);
   else {
    writes++;
    if(cache.GetCount() >= 0x7000) {
     removes++;
     cache.Remove(0, 100);
    }
    cache.Add(x, Fn(x));
   }
   rwlock.LeaveWrite();
  }
 }
}

CONSOLE_APP_MAIN
{
 Thread t[20];
 for(int i = 0; i < 9; i++)
  t[i].Run(callback1(WorkThread, i + 1));
 WorkThread(0);
 for(int i = 0; i < 9; i++)
  t[i].Wait();
}
```

This is basically a code to test RWMutex doing something reasonable - simulating cache.

This works as expected in Win32, fully utilizing both of my cores, but in Linux I am unable to get more than 60% CPU utilization. Obviously, some weird contention is involved, if only I would know why....

Any ideas?

Mirek