

HORROR IN THE “MUTEX” FOREST

Horror story starts in usual way.

One day I was interrupted by call, and the message was clear: Service is stopped, we cannot play bets (it is application which receives bet slips from internet web page). I immediately turn on my laptop.

Sound familiar.

OK. Program was running 2 weeks without interruption (this was the time of last update). What is wrong, now? It must be some Windows server update mambo jumbo. Let's restart application and go to lunch.

It was not first time that something unexplainable happen (not only to me). We all play in world of virtual machines. Maybe some admin change the disk online, or extend memory, change CPU.... Everybody, from time to time have this story. In today's world of horizontal scaling and many applications running on many environments this can happen, right. It is OK to FAIL.... After all: nodejs applications, java applications are well known they need restart from time to time. But, I'm not that kind of guy. I use FPC.

So what happen then? I went to lunch, order my meal, start to eat, and guess what: It happen again.

I got enough of it, It is time for investigation.

So, what I changed lately. Lets see little history of this application.

This is very complex multithread application with lot of memory which is shared between threads. Only one thread can write, others can read. It is http server which handles around 3 million requests per day.

I try to add new feature in application and I use FPC `TMultiReadExclusiveWriteSynchronizer` to ensure locks when reading and writing same piece of memory. When trying to add new function I found out that FPC `TMultiReadExclusiveWriteSynchronizer` suffers from some strange habit: When two readers try to read same memory, one reader blocks other. Ohh..., this is not what I expected, so I found some book and some other solution for the same thing. Book is *Tomes of Delphi Algorithms*. So, I adjusted the code from book, and since I use wrapper around `TMultiReadExclusiveWriteSynchronizer`, I just switch to new synchronizer. Normally, I did some test. Everything was OK, and I start application with new synchronizer. I did not use new functionality in application, I just switched to new synchronizer. And, bang, after 1 hour application blocks. Again, windows, update, antivirus.... you name it....

I put back the old application and it hangs again (luckily – after 5 hours). Definitely, it is not *Tomes of Delphi*...

WRAPPER

How can I found where is the bug in application. It must be bug. Probably some thread is waiting for MUTEX and I have dead lock somewhere. It is not only mutex in application. There is lot of them.

Since I use wrapper around sync object (I trust nobody, especially me), I decided to extend the wrapper function and count wait and release. I must have exact number of WaitForRead and ReleaseRead per thread, and the same story is for WaitForWrite and ReleaseWrite. How I can now where is the error. Instead of plain WaitForRead I rewrote it and it is now WaitForRead(InLocation: string). Same for other three functions, of course.

On top of that I start one thread which will dump every second status of all sync objects and its thread access. It took me two days to complete code, make some tests, make some false tests and I was ready to change my service. This took me another 2 days (thanks to compiler, this was easy and possible). After 4 days I start the program, and this was it.

Program wrote in error file the location of the error. And the error was.

WaitForRead

WaitForRead

Do some stuff....

ReleaseRead

ReleaseRead

You cannot do that on kind of MUTEX that I have. It was not REENTRY MUTEX. Actually, the problem was not MUTEX, it was SEMAPHORE after MUTEX which was not released in sync object. So, why Toms Of Delphi stops very quickly, and FPC TMultiReadExclusiveWriteSynchronizer stops sometime or never. I did not investigate too much TMultiReadExclusiveWriteSynchronizer, but it seems that it is much more flexible on that issue. Toms Of Delphi sync is clear: It creates Semaphore with limited amount of possible readers. This amount was reached, and program stops and hangs, waiting forever.

This bug was 2 years old in application, and finally it was caught.

Source code, and test program for Windows and Linux is in zip file. Compiler used is from Pilot Logic Code Typhon 6.00.

If you have the time to play with code and change from FPC to TomeOfDephi synchronizer, you will see that the other one is faster (Line 375 in FavSemaphore)

```
//TFavWriterReaderSemaphoreWriterPriority = TFavWriterReaderSemaphoreSimple;  
//TFavWriterReaderSemaphoreWriterPriority = TFavWriterReaderSemaphoreWriterPriorityFPC;  
TFavWriterReaderSemaphoreWriterPriority = TFavWriterReaderSemaphoreWriterPriorityTomesOfDelphi;
```

Try to change number of writers and readers to see the difference.

CONST

CONST_READER_COUNT = 16; // NUMBER OF READER THREADS

CONST_WRITER_COUNT = 2; // NUMBER OF WRITER THREAD

CONST_READER_SLEEP = 1; // LENGTH IN MS OF READER JOB

CONST_WRITER_SLEEP = 10; // LENGTH IN MS OF WRITER JOB

The difference is especially visible when you have 2 or more writers.

Program will create SyncObjects.dmp so you can see the status of your mutexes, and SyncObjects.err if you have error in your code

Sorry for my English. This is not the language my mother speaks.

DINKO MILJAK

dinmil@gmail.com

www.leapbit.com