Checking DLLs for Thread Safety

Introduction

There might be situation where you need to check if the DLLs provided by 3rd party is thread safe or not. This can be done by writing a short test driver (given below) to encompass DLL calls with OpenMP directives. The entire application is then analyzed with Intel® Thread Checker and checked to see if there are any errors. Please note this test is not a guarantee of thread safety, but it is an excellent test when source is available and the only test we know of when source is not available.

DLL

Let us consider the PrimesDLL which returns the number of primes within the range that are mentioned as arguments.

Entire DLL code

```
extern "C" long FindPrimes(long start, long end, long skip, CALLBACK_FUNC func)
{
    gPrimesFound = 0;
    for( int i = start; i <= end; i += skip )
    {
        if( TestForPrime(i) )
            gPrimesFound++;

            (*func)( i );
    }

    return gPrimesFound;

}

static bool TestForPrime(int val)
{
    int limit, factor = 3;

    limit = (long)(sqrtf((float)val)+0.5f);
    while( (factor <= limit) && (val % factor))
        factor ++;

    return (factor > limit);

}
```

Test Driver

The test driver calls the DLL twice in each of the section. You can replace the FindPrimes with the DLL function you are interested in.

PrimeDLL test driver:

```
 PrimeData test[2];
......
#pragma omp parallel sections num_threads(2)
{
   #pragma omp section
      primesFound1 = FindPrimes(test[0].start, test[0].end, ...);
   #pragma omp section
     primesFound2 = FindPrimes(test[1].start, test[1].end, ...);
}
```

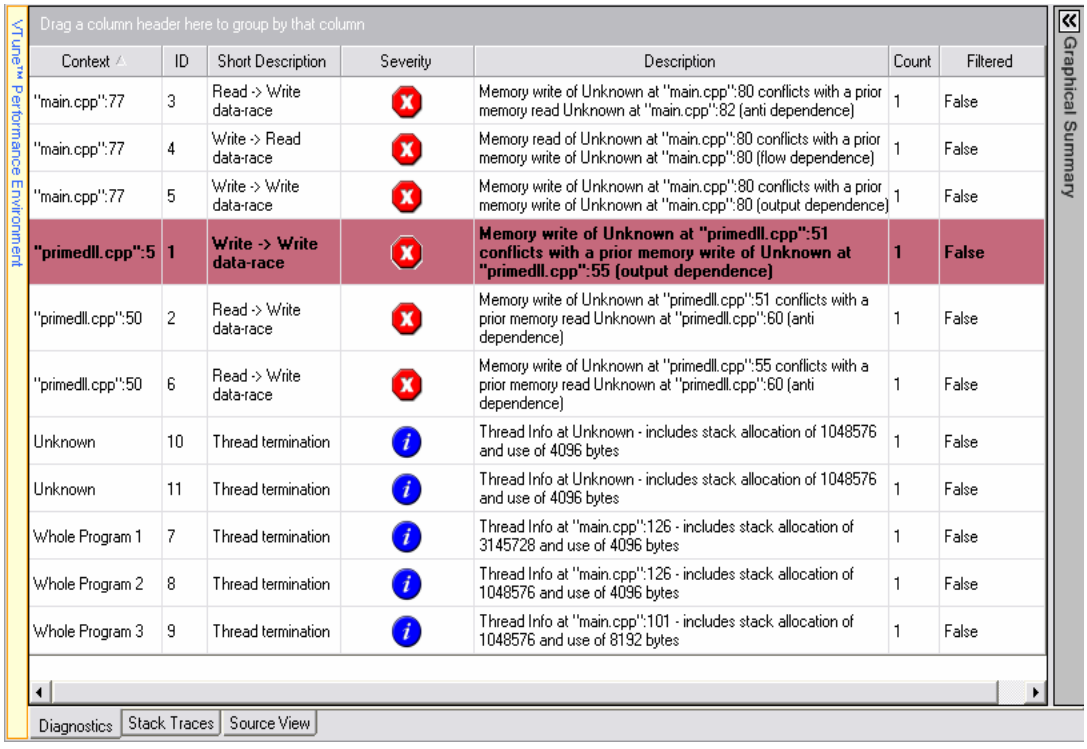
Thread Checker Analysis

The test driver was run with the Intel Thread Checker and it points out threading issues related to data-races, deadlocks and stalls.

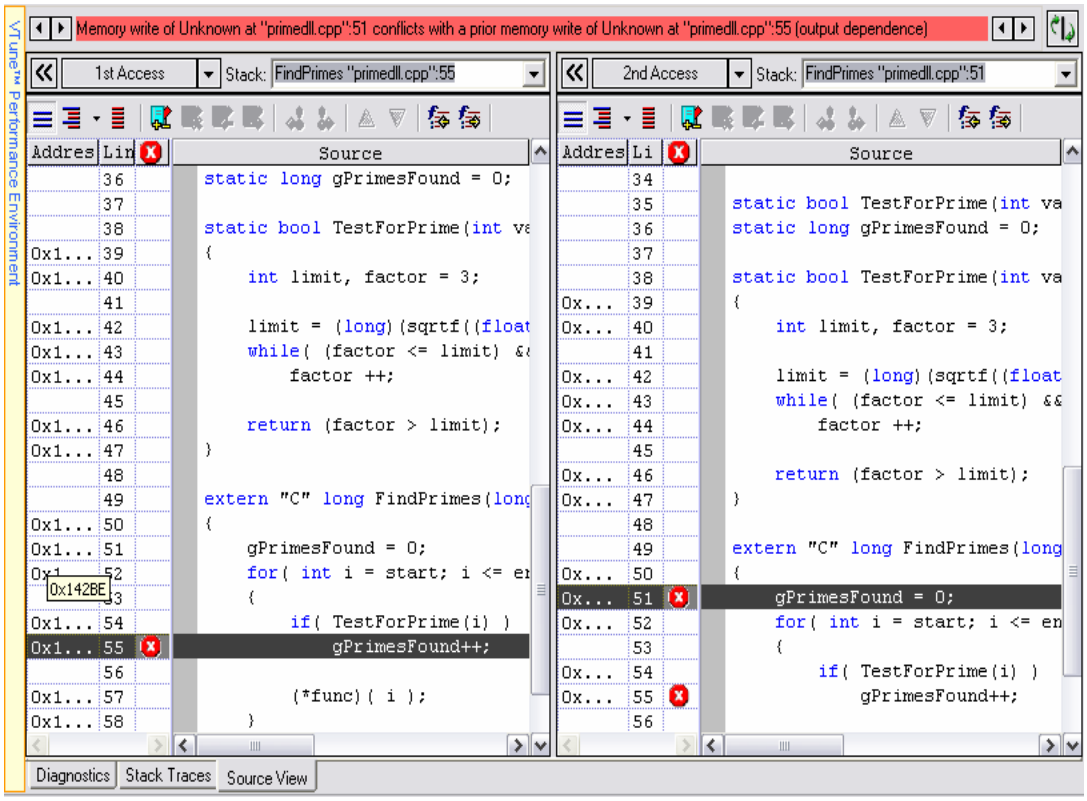
If Intel® Thread Checker only reports informational issues (blue circles with an 'i' inside), it means Intel® Thread Checker did not identify any issues indicating the dll is not thread safe. Intel® Thread Checker only checks routines exercised by the data. You may want to vary the data set to exercise additional code path and repeat.

If Intel® Thread Checker identifies issues with red (hexagon) or yellow (polygon) then this indicates the dll is not thread safe. If source is available you can double click to go to source view. If source is not available you can double click to see a call stack view of where issue occurred.

The output below shows Write->Write data races in the FindPrimes function in the DLL code. If you right click on the source code, it pinpoints to the exact line in the code where there is a memory conflict.



Context	ID	Short Description	Severity	Description	Count	Filtered
"main.cpp":77	3	Read -> Write data-race		Memory write of Unknown at "main.cpp":80 conflicts with a prior memory read Unknown at "main.cpp":82 (anti dependence)	1	False
"main.cpp":77	4	Write -> Read data-race		Memory read of Unknown at "main.cpp":80 conflicts with a prior memory write of Unknown at "main.cpp":80 (flow dependence)	1	False
"main.cpp":77	5	Write -> Write data-race		Memory write of Unknown at "main.cpp":80 conflicts with a prior memory write of Unknown at "main.cpp":80 (output dependence)	1	False
"primedll.cpp":5	**1**	**Write -> Write data-race**		**Memory write of Unknown at "primedll.cpp":51 conflicts with a prior memory write of Unknown at "primedll.cpp":55 (output dependence)**	**1**	**False**
"primedll.cpp":50	2	Read -> Write data-race		Memory write of Unknown at "primedll.cpp":51 conflicts with a prior memory read Unknown at "primedll.cpp":60 (anti dependence)	1	False
"primedll.cpp":50	6	Read -> Write data-race		Memory write of Unknown at "primedll.cpp":55 conflicts with a prior memory read Unknown at "primedll.cpp":60 (anti dependence)	1	False
Unknown	10	Thread termination		Thread Info at Unknown - includes stack allocation of 1048576 and use of 4096 bytes	1	False
Unknown	11	Thread termination		Thread Info at Unknown - includes stack allocation of 1048576 and use of 4096 bytes	1	False
Whole Program 1	7	Thread termination		Thread Info at "main.cpp":126 - includes stack allocation of 3145728 and use of 4096 bytes	1	False
Whole Program 2	8	Thread termination		Thread Info at "main.cpp":126 - includes stack allocation of 1048576 and use of 4096 bytes	1	False
Whole Program 3	9	Thread termination		Thread Info at "main.cpp":101 - includes stack allocation of 1048576 and use of 8192 bytes	1	False


VTune™ Performance Environment
Memory write of Unknown at "primedll.cpp":51 conflicts with a prior memory write of Unknown at "primedll.cpp":55 (output dependence)
1st Access
Stack: FindPrimes "primedll.cpp":55
2nd Access
Stack: FindPrimes "primedll.cpp":51
Addres
Lin
Source
0x142BE
Diagnostics
Stack Traces
Source View