Debugging Serial and OpenMP Programs with Visual Studio 2008

Christian Terboven

terboven@rz.rwth-aachen.de

Center for Computing and Communication

RWTH Aachen University

PPCES 2009

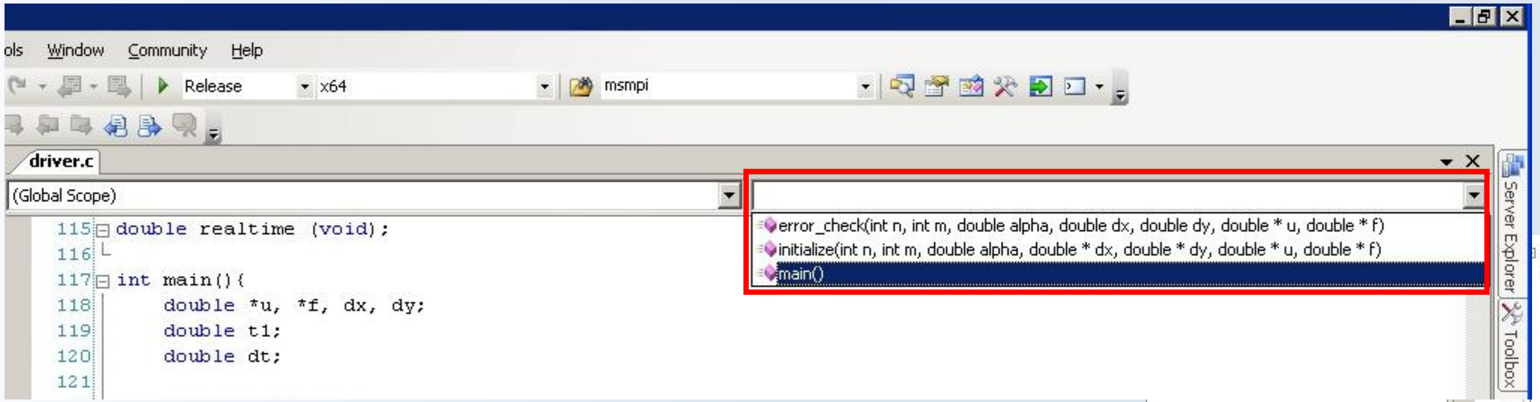
March 23rd, RWTH Aachen University

Agenda

- Debugging Serial Programs
- Debugging OpenMP Programs
- Debugging OpenMP Program w/ DDTlite
- Demo

Source navigation in Visual Studio

- Navigating a C/C++ project:

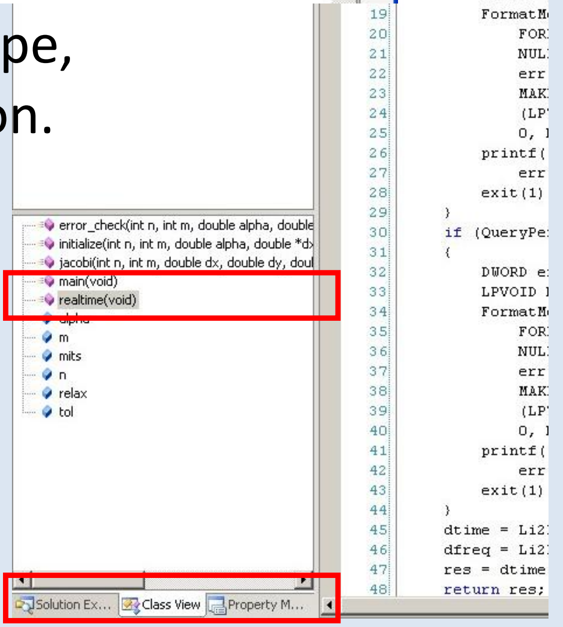


```
double realtime (void);

int main(){
    double *u, *f, dx, dy;
    double t1;
    double dt;
```

```
error_check(int n, int m, double alpha, double dx, double dy, double * u, double * f)
initialize(int n, int m, double alpha, double * dx, double * dy, double * u, double * f)
main()
```

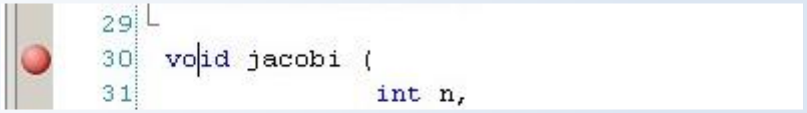
  - By choosing a function of the current scope, the cursor jumps to the function definition.
  - You can use the class view to navigate through your code as well.


error_check(int n, int m, double alpha, double
initialize(int n, int m, double alpha, double *dx
jacobi(int n, int m, double dx, double dy, doul
main(void)
realtime(void)
alpha
m
mits
n
relax
tol
Solution Ex...
Class View
Property M...

```
double realtime
{
    LARGE_INTEG
    double dtim
    if (QueryPe
    {
        DWORD e
        LPVOID l
        FormatM
            FOR
            NUL
            err
            MAK
            (LP
            0, 
        printf(
            err
        exit(1)
    }
    if (QueryPe
    {
        DWORD e
        LPVOID 
        FormatM
            FOR
            NUL
            err
            MAK
            (LP
            0, 
        printf(
            err
        exit(1)
    }
    dtime = Li2
    dfreq = Li2
    res = dtime
    return res;
```

Debugging Serial Programs (1/2)

- A breakpoint can be set by clicking in the grey area left of the line number. Clicking again removes the breakpoint.



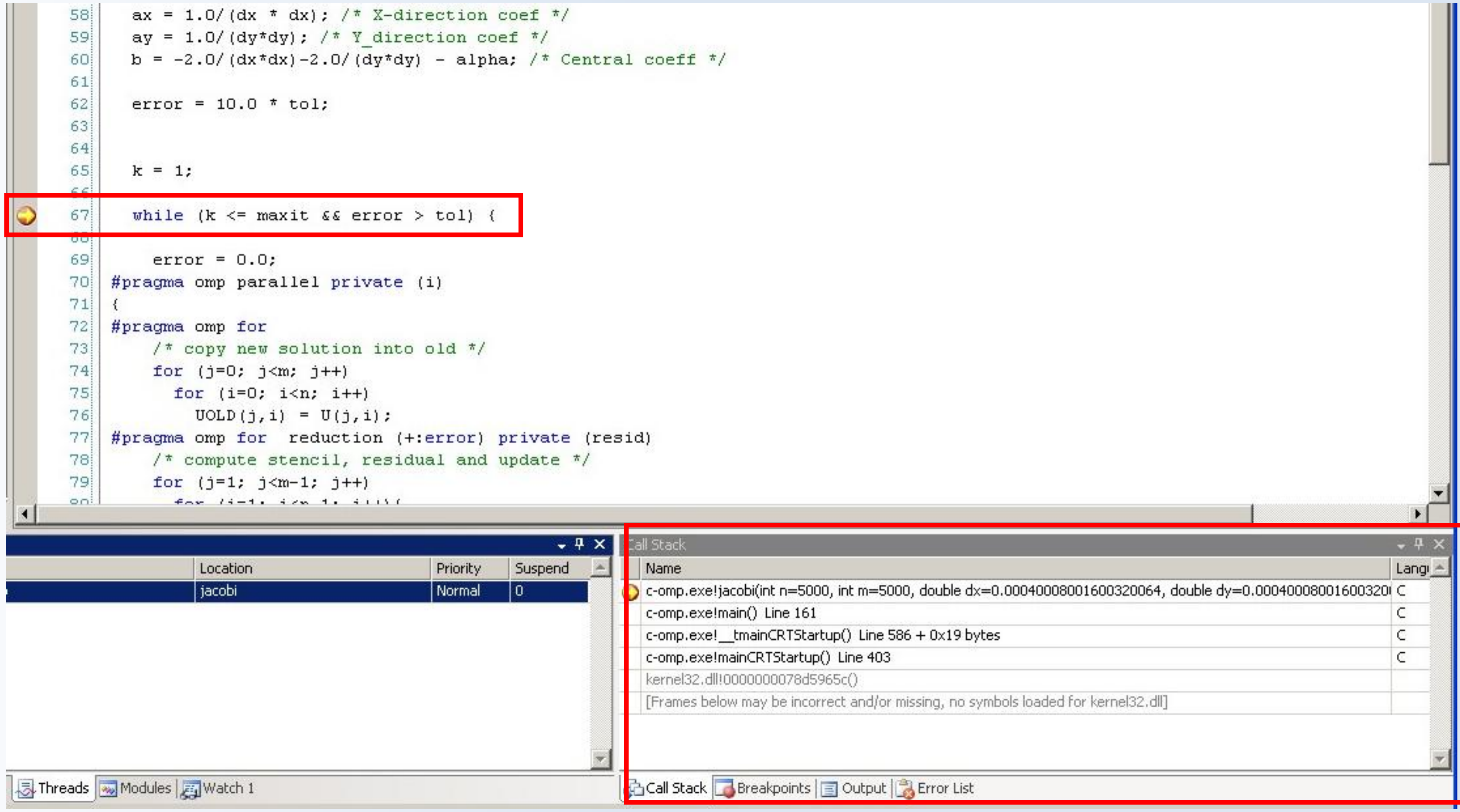
```
void jacobi (
                int n,
```

- Right-clicking on a breakpoint shows the context menu with the following functions
  - Disable a breakpoint (temporary)
  - Set breakpoint trigger conditions
  - Trigger filter for selected threads or processes
  - Define actions to be executed when the breakpoint is triggered

- Just hold the mouse over a variable for a short moment to get the actual value displayed (*hover*). This is also possible for marked expression (to some extent).

Debugging Serial Programs (2/2)

- During a debugging session, the actual program location is marked by a yellow arrow. You can drag this arrow up/down.



```
    ax = 1.0/(dx * dx); /* X-direction coef */
    ay = 1.0/(dy*dy); /* Y_direction coef */
    b = -2.0/(dx*dx)-2.0/(dy*dy) - alpha; /* Central coeff */

    error = 10.0 * tol;


    k = 1;

    while (k <= maxit && error > tol) {

      error = 0.0;
#pragma omp parallel private (i)
{
#pragma omp for
      /* copy new solution into old */
      for (j=0; j<m; j++)
        for (i=0; i<n; i++)
          UOLD(j,i) = U(j,i);
#pragma omp for  reduction (+:error) private (resid)
      /* compute stencil, residual and update */
      for (j=1; j<m-1; j++)
```

Location	Priority	Suspend
jacobi	Normal	0

Threads | Modules | Watch 1

Call Stack

Name	Lang
c-omp.exe!jacobi(int n=5000, int m=5000, double dx=0.00040008001600320064, double dy=0.000400080016003200	C
c-omp.exe!main() Line 161	C
c-omp.exe!__tmainCRTStartup() Line 586 + 0x19 bytes	C
c-omp.exe!mainCRTStartup() Line 403	C
kernel32.dll!0000000078d5965c()	
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]	

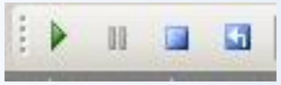
Call Stack | Breakpoints | Output | Error List

Agenda

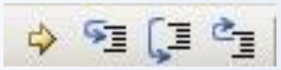
- Debugging Serial Programs
- Debugging OpenMP Programs
- Debugging OpenMP Program w/ DDTlite
- Demo

Debugging OpenMP Programs (1/3)

- Debugging of OpenMP applications in Visual Studio 2008 works with all compilers: the Microsoft C/C++ compiler, the Intel C/C++ compiler and the Intel Fortran compiler.
- Note: If you use the Intel compiler and let a program run with `n` threads, you will see `n+1` threads (one management thread).
- We advise you to compile without any optimization for debugging, that means use the pre-configured *Debug* configuration and just enable OpenMP.
- Control debugging:



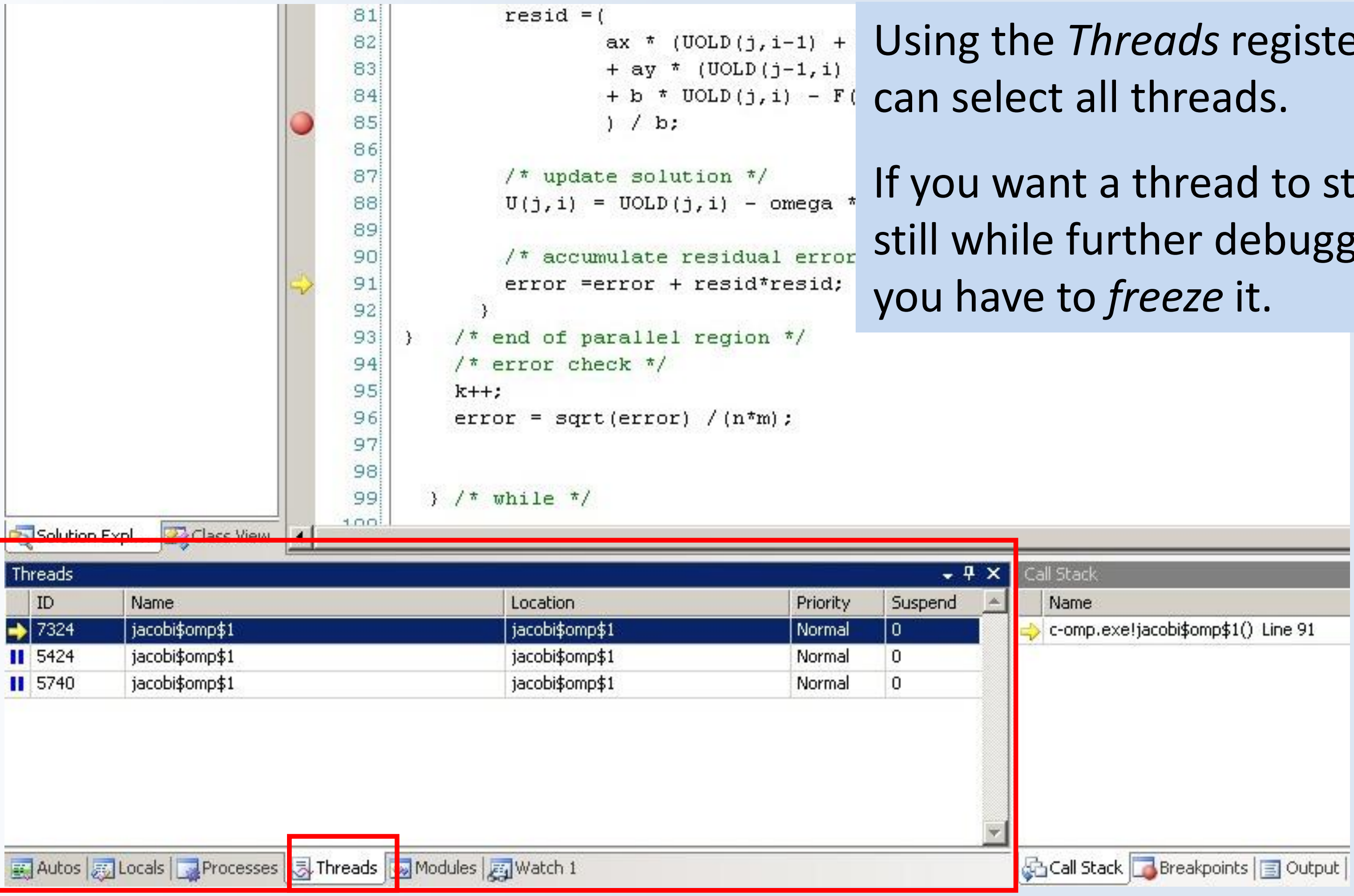
» Start / Continue, Break, Stop, Restart



» Show next statement, Step Into, Step Over, Step Out

Debugging OpenMP Programs (2/3)

- All threads stop at a breakpoint (first thread encounters it).



```
         resid =(
                 ax * (UOLD(j,i-1) +
                 + ay * (UOLD(j-1,i)
                 + b * UOLD(j,i) - F(
                 ) / b;

         /* update solution */
         U(j,i) = UOLD(j,i) - omega *

         /* accumulate residual error
         error =error + resid*resid;
       }
}    /* end of parallel region */
     /* error check */
     k++;
     error = sqrt(error) /(n*m);


  } /* while */
```

Threads

	ID	Name	Location	Priority	Suspend
➡	7324	jacobi$omp$1	jacobi$omp$1	Normal	0
❚❚	5424	jacobi$omp$1	jacobi$omp$1	Normal	0
❚❚	5740	jacobi$omp$1	jacobi$omp$1	Normal	0

Call Stack

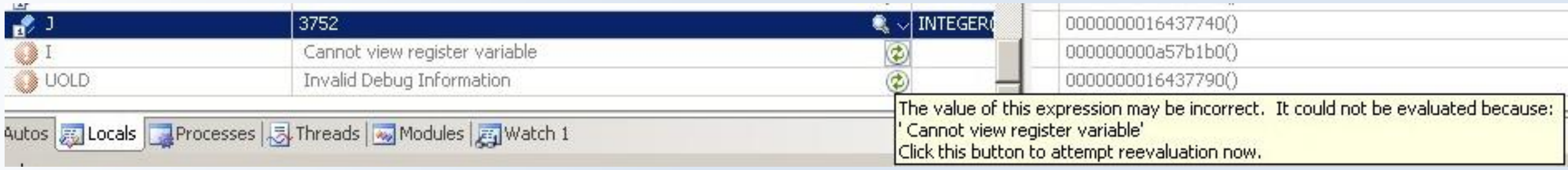
Name
c-omp.exe!jacobi$omp$1() Line 91

Using the *Threads* register, you can select all threads.

If you want a thread to stand still while further debugging, you have to *freeze* it.

Debugging OpenMP Programs (3/3)

- For all threads, you can view the *local* and *shared* variables.
  - The *Locals* register contains all variables of the current scope.
  - The *Autos* register contains a set of interesting variables guessed by the compiler – remarkably good.
- Some limitations when using the Intel Fortran compiler:
  - The *Autos* register is empty, *Locals* is working fine.
  - One can not (at least sometimes) identify the management thread by the name – it is the one you get an error message of no source code being available if you select it ;-)
  - Sometimes expressions have to be updated because of (possible) compiler optimization.



J	3752	INTEGER	0000000016437740()
I	Cannot view register variable		000000000a57b1b0()
UOLD	Invalid Debug Information		0000000016437790()

The value of this expression may be incorrect. It could not be evaluated because:
' Cannot view register variable'
Click this button to attempt reevaluation now.

Agenda

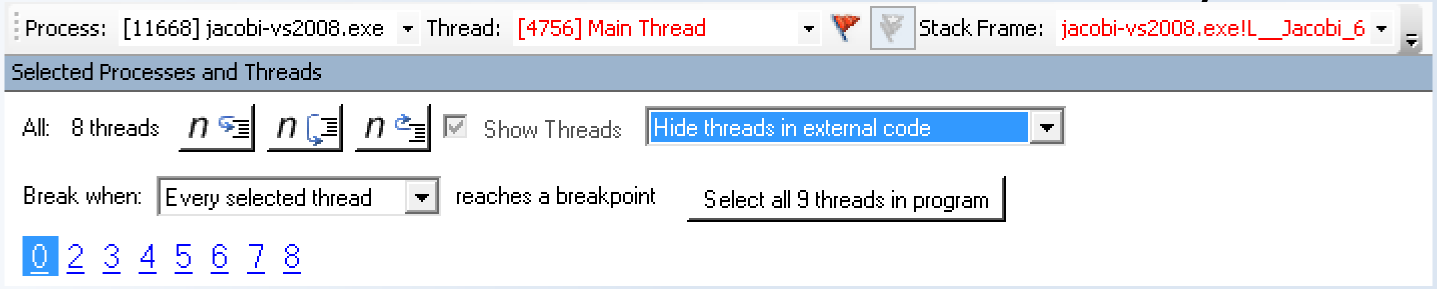
- Debugging Serial Programs
- Debugging OpenMP Programs
- Debugging OpenMP Program w/ DDTlite
- Demo

Allinea DDTlite Overview

- DDTlite is a plugin for Visual Studio 2008 SP1 (and newer)
  - Mainly marketed for improved MPI debugging experience
  - But also improves OpenMP debugging experience
- Adds the following features:
  - Control processes and threads individually, grouped or together
  - Examine variable values per thread / process
  - Parallel Stack View per thread / process
  - Location View per thread / process
  - Thread / process group management
- Available on our `cluster-win-beta` and `-lab` frontends
- Note: We recommend using the Microsoft Compilers when using DDTlite...

Debugging OpenMP Programs w/ DDTlite

- Select and switch between threads individually:



Process: [11668] jacobi-vs2008.exe Thread: [4756] Main Thread Stack Frame: jacobi-vs2008.exe!L__Jacobi_6

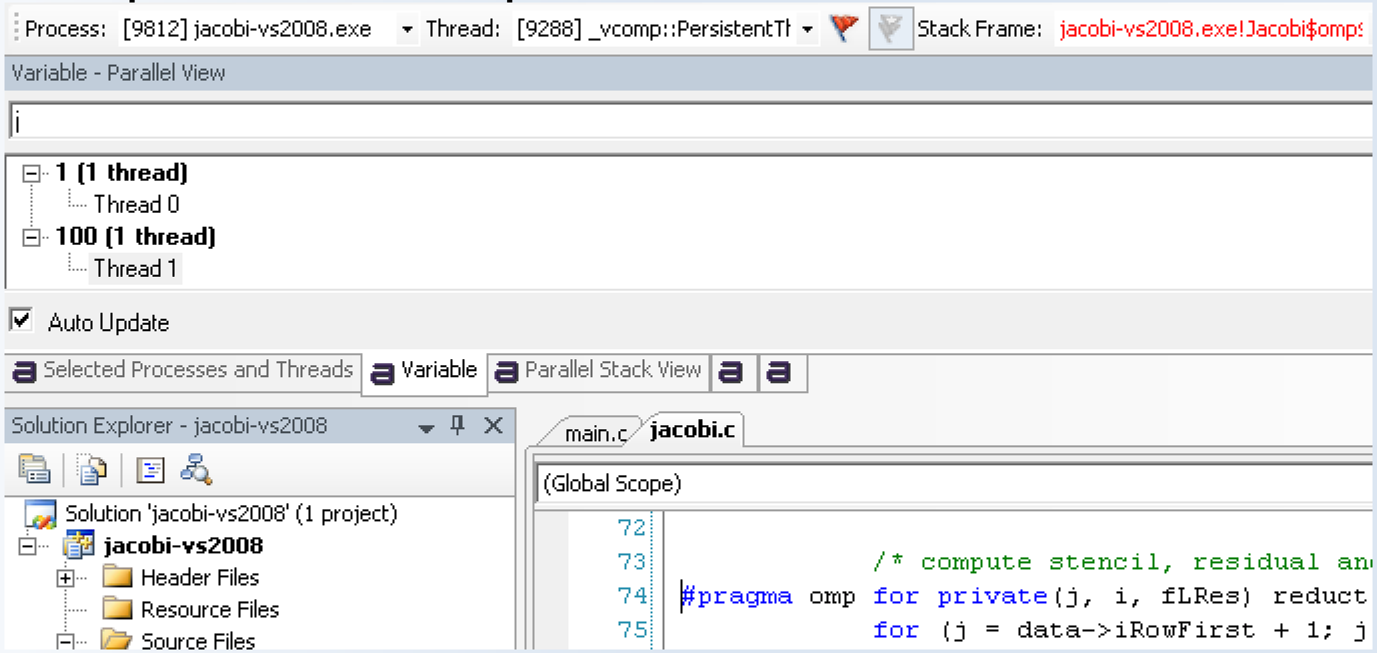
Selected Processes and Threads

All: 8 threads Show Threads Hide threads in external code

Break when: Every selected thread reaches a breakpoint Select all 9 threads in program

0 2 3 4 5 6 7 8

- Compare variables per thread:



Process: [9812] jacobi-vs2008.exe Thread: [9288] _vcomp::PersistentTl Stack Frame: jacobi-vs2008.exe!Jacobi$omp

Variable - Parallel View

i

```
1 (1 thread)
    Thread 0
100 (1 thread)
    Thread 1
```

Auto Update

```
            /* compute stencil, residual an
#pragma omp for private(j, i, fLRes) reduct
            for (j = data->iRowFirst + 1; j
```

Agenda

- Debugging Serial Programs
- Debugging OpenMP Programs
- Debugging OpenMP Program w/ DDTlite
- Demo

Demo

Jacobi-C-OMP

The End

Thank you for your attention!

Questions?