RWTH AACHEN UNIVERSITY

Visual Studio 2010

Best Practices

Christian Terboven <terboven@rz.rwth-aachen.de>

19.03.2012 / Aachen, Germany

Stand: 16.03.2012

Version 2.3

Agenda

- Extensions
- Project Templates
- Portable Time Measurement
- Using Solutions and Projects
- Fixing an example application: LBM-2D (Joppich)

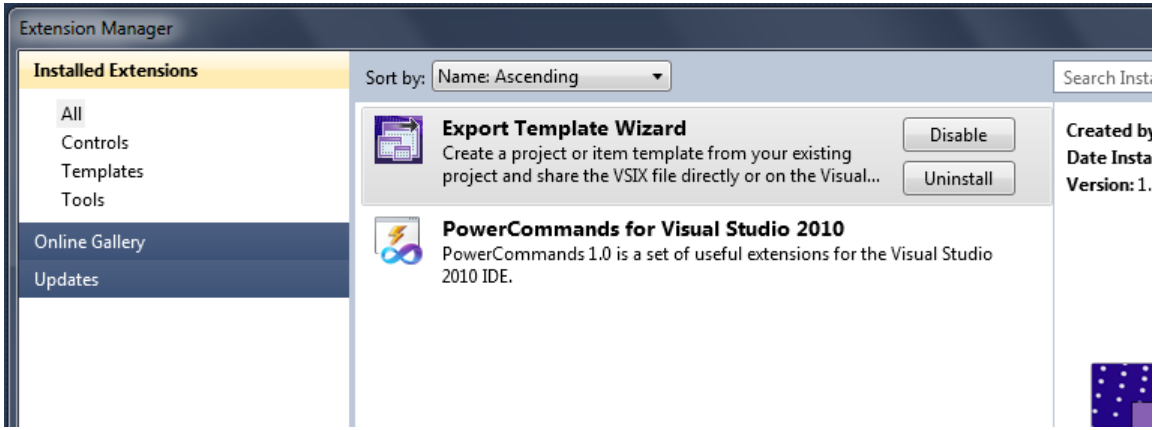
Extensions

Visual Studio 2010 Extensions

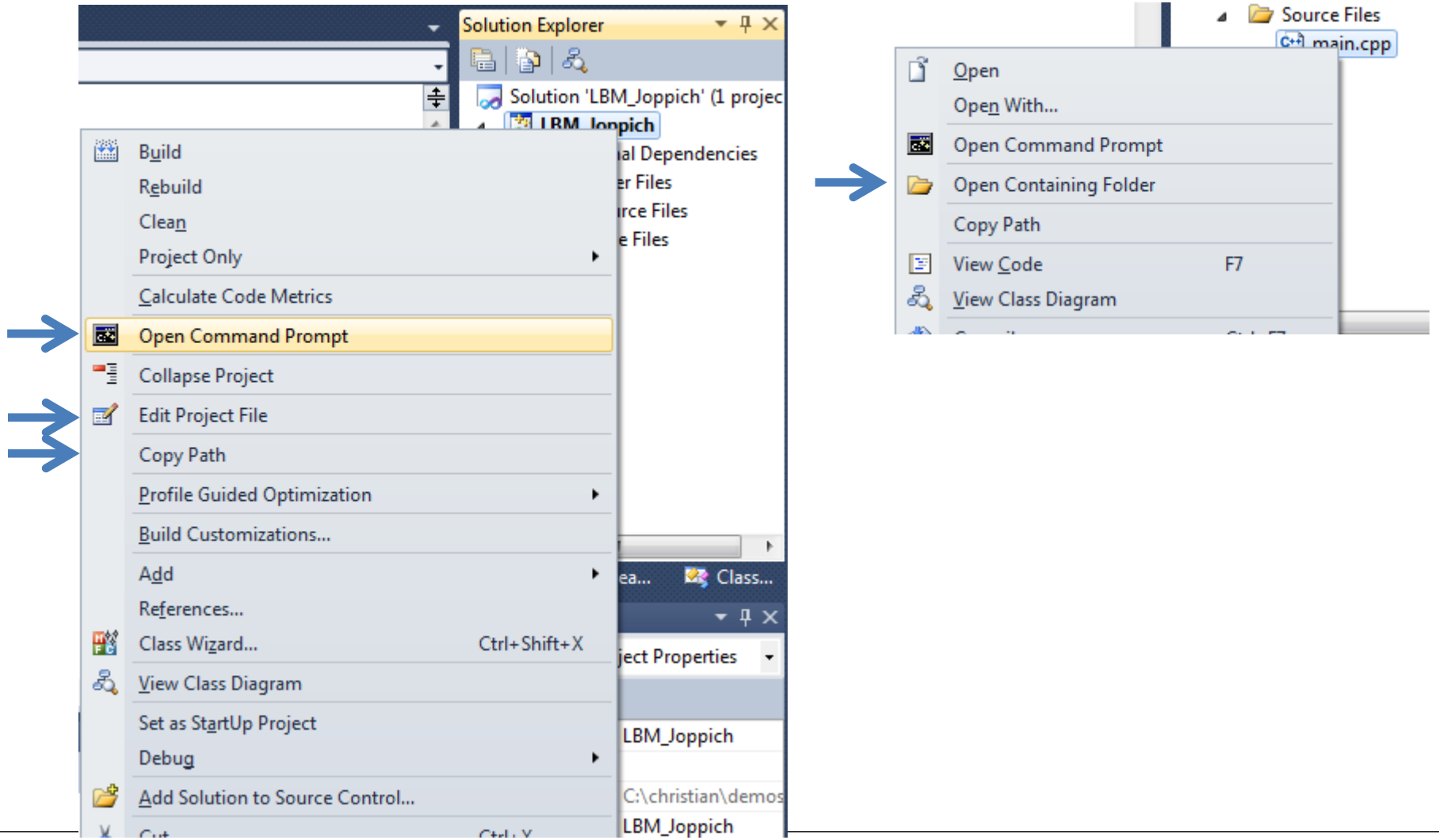
- **Visual Studio 2010 introduced the Extension Gallery: http://visualstudiogallery.msdn.microsoft.com/en-us/**

- **The list of available extensions is still growing, here I would like to present just one extension:**
  - *PowerCommands for Visual Studio 2010*: Several useful extensions.

- **Tools → Extension Manager… → Online Gallery → Use the search box to find what you are looking for:**



- **Adds several useful context menu items:**



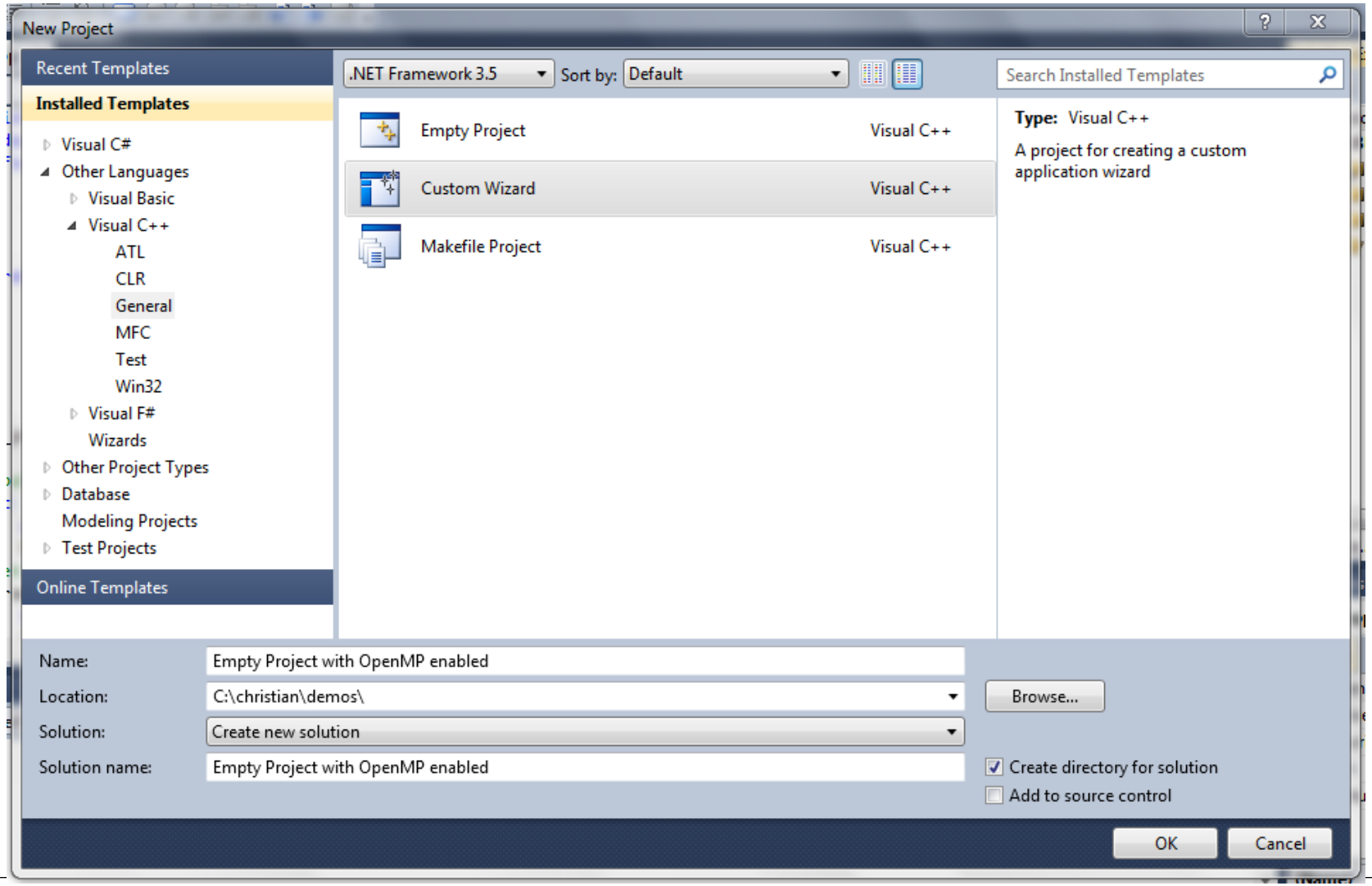
Project Templates

Project Template Types

- **There are two different types of project templates:**
  - Visual C++
  - Everything else
- **For *Everything Else* it is pretty simple to create a project template:**
  - Create a solution+project of suitable type
  - Modify the settings accordingly to your idea
  - Select File → Export Template… and follow the wizard
  - You may also get the *Export Template Wizard* extensions for more comfort
- **For *Visual C++* projects this menu item is either greyed out or does not offer interesting options**
  - Use the *Custom Wizard* instead

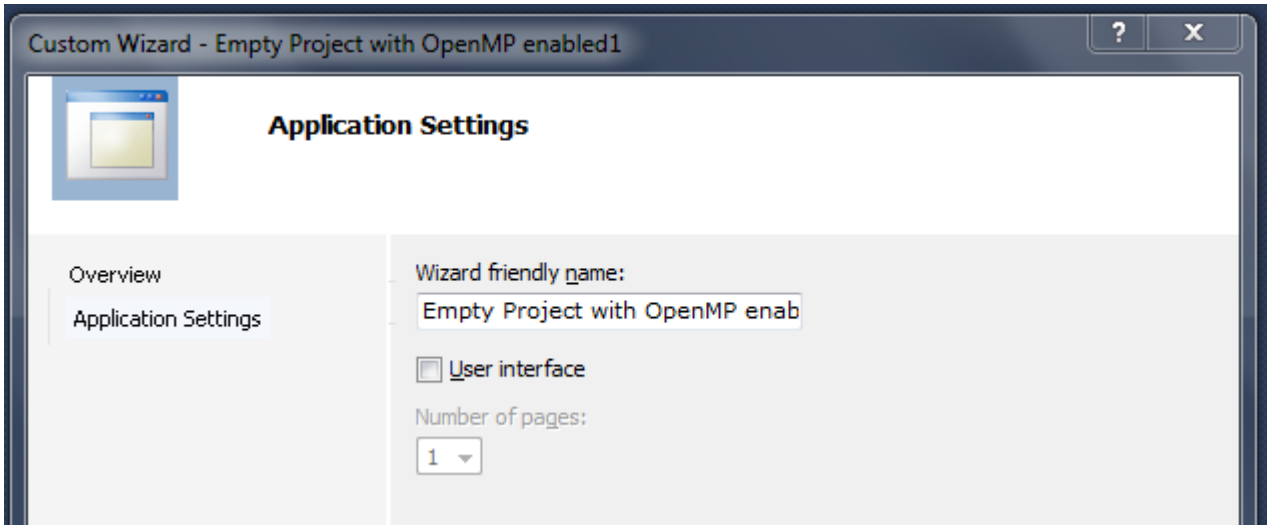
Visual C++ Custom Wizard (1/3)

- The *Custom Wizard* allows to build project templates of all sorts



Visual C++ Custom Wizard (2/3)

- **Question: Do you want to design a wizard UI, or just some settings:**



- **Function *AddConfig()* in script file *default.js* is very important:**

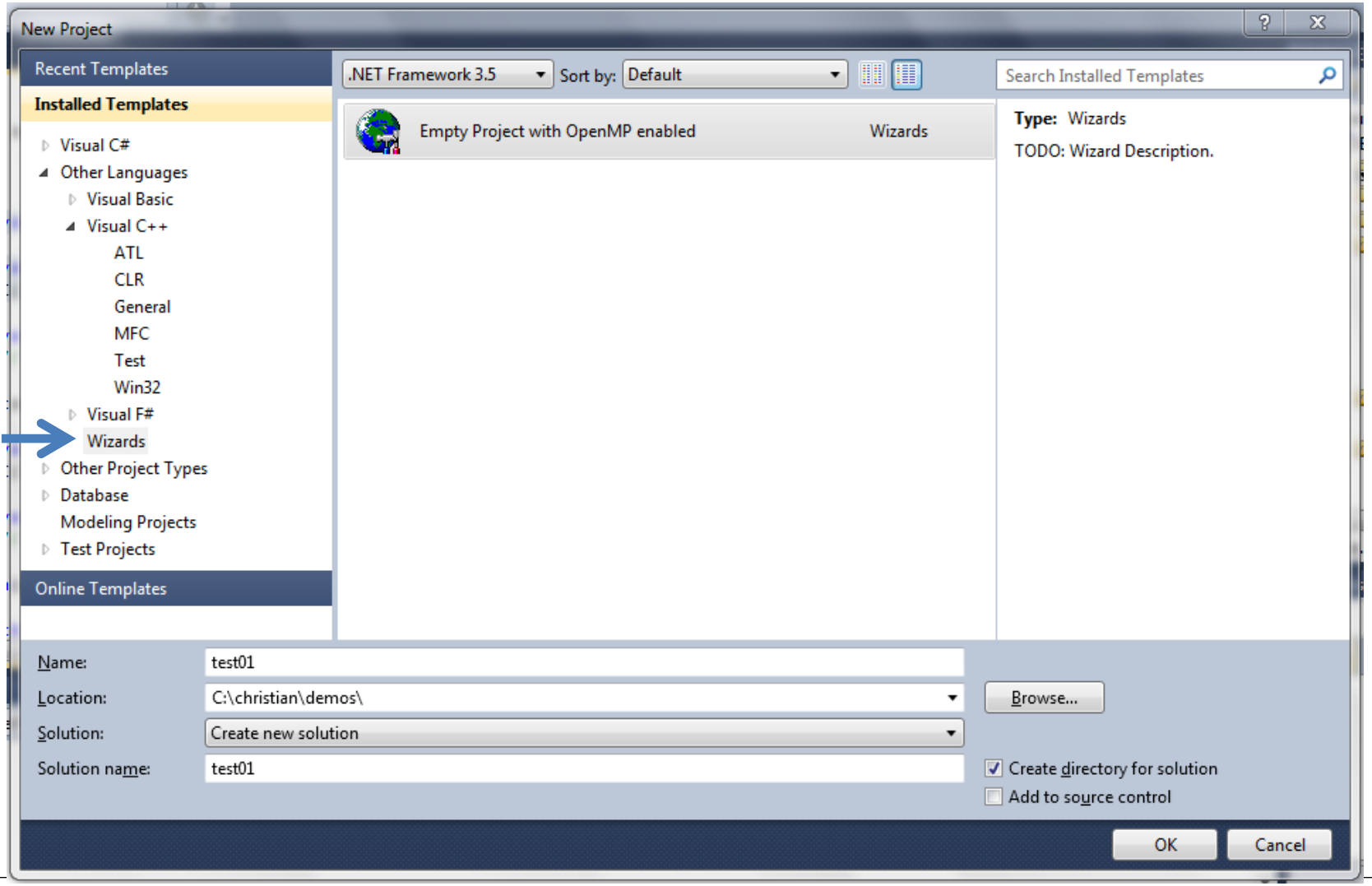
```
var config = proj.Object.Configurations('Debug');
var CLTool = config.Tools('VCCLCompilerTool');
CLTool.OpenMP = true;
```

  - Enables OpenMP-aware compilation in the Debug configuration

- **All details on the object model:**
  http://msdn.microsoft.com/en-us/library/ms168842.aspx

Visual C++ Custom Wizard (3/3)

- **Your wizard will be found here:**



Portable Time Measurement

Portable Time Measurement (1/3)

- **Porting applications from Unix to Windows (or the other way around) can be quite hard … but it was not for most user codes (HPC) we tried on Windows.**

  (1) The most common problem was time measurement as `gettimeofday()` is not available on Windows,

  (2) followed by directory management issues where ‚/‘ instead of ‚\‘ had been used before.

- **In most cases we addressed (2) using `#ifdefs`.**
- **Handling (1) depends on the programming language:**
  - C++: We have written a version of `double realtime()` for Windows and Unix.
  - FORTRAN: As the library (defined along with the language) already provides time measurement facilities, we used these.

Portable Time Measurement (2/3)

```
#ifdef WIN32
   #include <Windows.h>
   #define Li2Double(x) ((double)((x).HighPart) * 4.294967296E9 + \
      (double)((x).LowPart))
#else
  #include <sys/time.h>
  #include <time.h>
#endif

double realtime (void) {
#ifdef WIN32
   LARGE_INTEGER time, freq;
   double dtime, dfreq;
   if (QueryPerformanceCounter(&time) == 0) { … error … }
   if (QueryPerformanceFrequency(&freq) == 0) { … error … }
   return Li2Double(time) / Li2Double(freq);
#else
   struct timeval tv;
   gettimeofday(&tv, (struct timezone*)0);
   return ((double)tv.tv_sec + (double)tv.tv_usec / 1000000.0 );
}
```

Portable Time Measurement (3/3)

RWTH AACHEN UNIVERSITY

Taking time the MPI way:

```
#include <mpi.h>
    ...
   double t1, t2, elapsed_seconds;
   t1 = MPI_Wtime();
   ...
   t2 = MPI_Wtime();
   elapsed_seconds = t2 - t1;
```

Taking time the OpenMP way:

```
#include <omp.h>
    ...
   double t1, t2 elapsed_seconds;
   t1 = omp_get_wtime();
   ...
   t2 = omp_get_wtime();
   elapsed_seconds = t2 - t1;
```

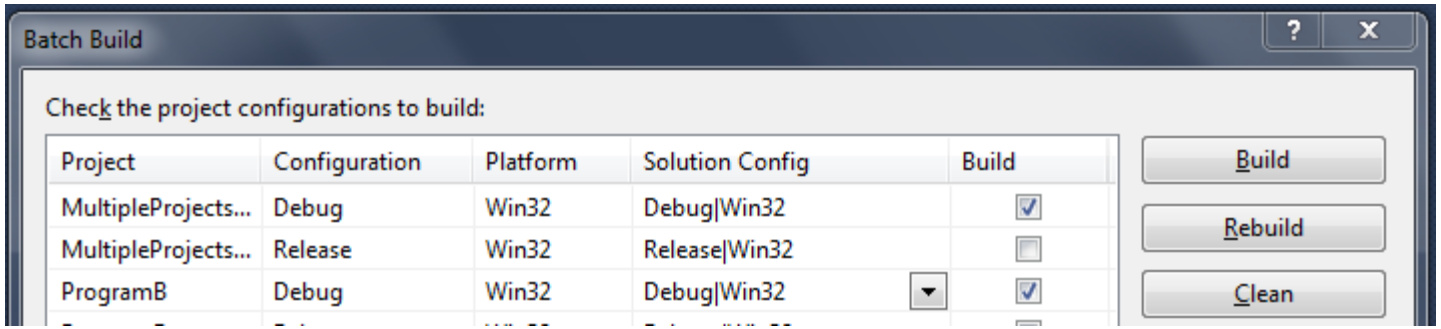
Using Solutions and Projects

… for Interactive Demos

- **There are several advantages of having multiple (small) Projects combined in one single Solution:**
  - The ability to build projects or the whole solution in an automatic fashion, i.e. to check a modified system configuration.

```
devenv  solution.sln  /build  [configuration]  [/project  projectname]
devenv  solution.sln  /clean  [...]
devenv  solution.sln  /run  [...]
devenv  solution.sln  /useenv  command={build|clean|run}
```

  - Batch Build vial Build → Batch Build …



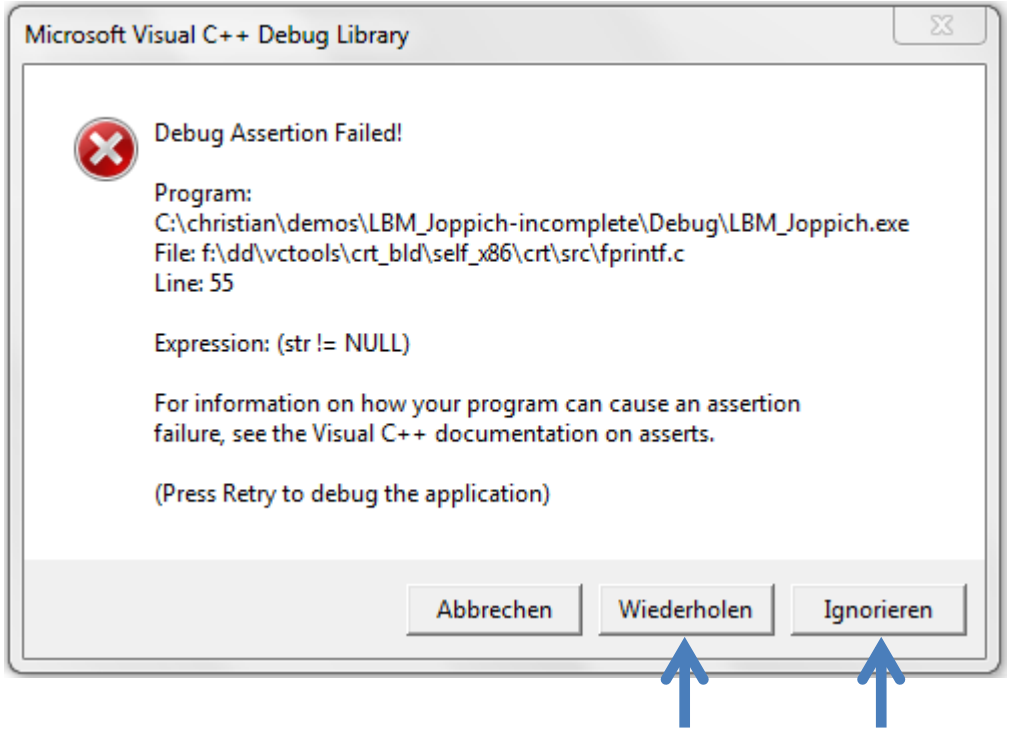
  - Select the *active* project by right-clicking the project and Set as StartUp Project

Fixing an example application: LBM-2D (Joppich)

Getting it to work … (1/9)

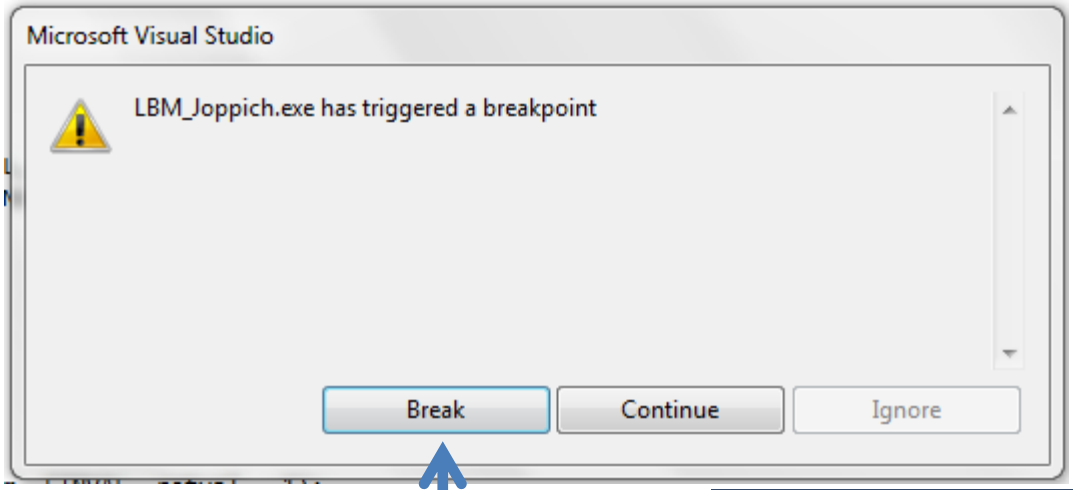
- **Recommendation: Always use *Debug* configuration until you know the program really works fine.**

- **Step 1: Run the program via Debug → Start Debugging. Within a few seconds an assertion failed:**

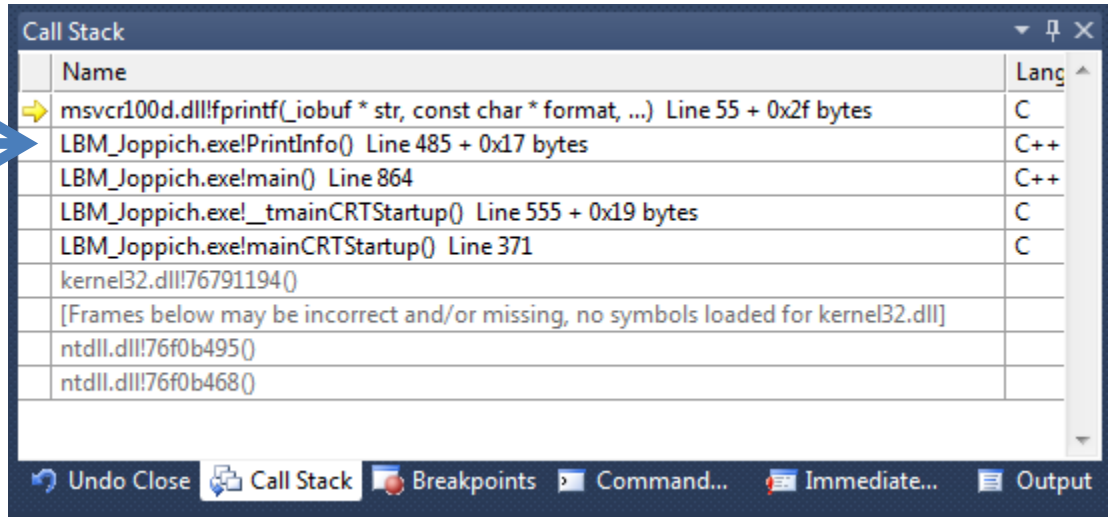


Getting it to work … (2/9)

- Thanks to the Debug-runtime the Visual Studio Debugger gets control:



- Break the program execution and use the Call Stack window to find out where the error occured.



Getting it to work … (3/9)

- This is the user code …

```
477  // SimulationsParametern
478  void
479  PrintInfo()
480  {
481      // Erstellt ein M-File fuer Matlab mit SimulationsParametern
482      FILE *fp;
483        fp = fopen("LBMData2D/LBMinfo2D.m", "w+");
484    //    fp = fopen("LBMinfo2D.m", "w+");
485        fprintf(fp,"%i\n", LATTICE_WIDTH_X);
```

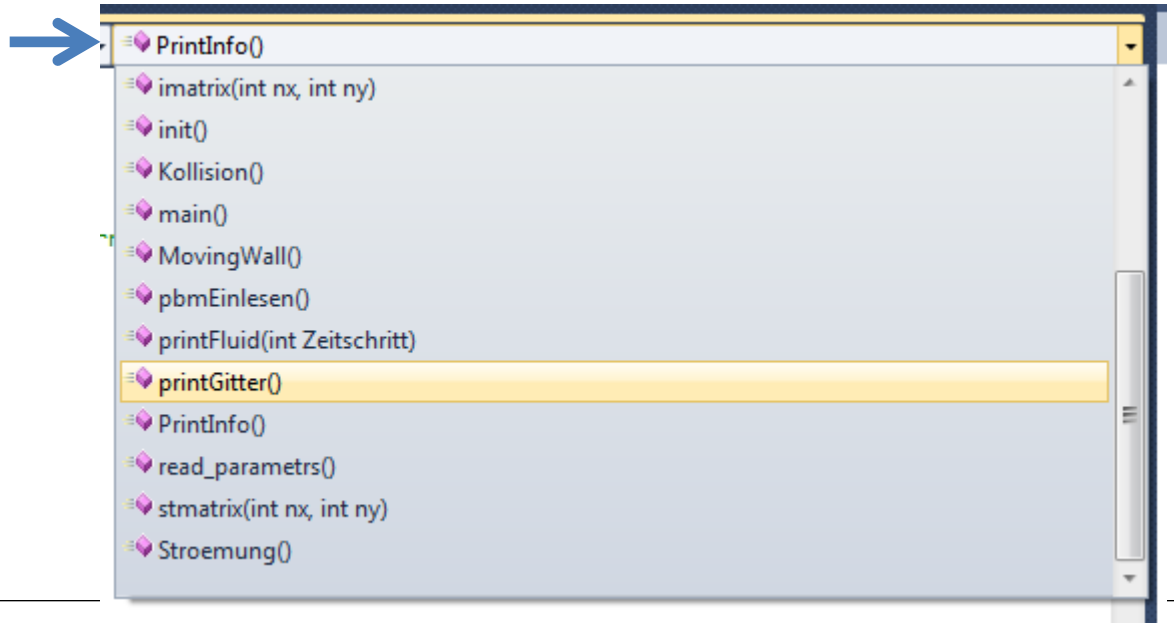
- … and this is the fprintf() implementation …

```
41   int __cdecl fprintf (
42           FILE *str,
43           const char *format,
44           ...
45           )
46   /*
47    * 'F'ile (stream) 'PRINT', 'F'ormatted
48    */
49   {
50       va_list(arglist);
51       REG1 FILE *stream;
52       REG2 int buffing;
53       int retval=0;
54
55       _VALIDATE_RETURN( (str != NULL), EINVAL, -1);
```

- Two issues: (i) subdirectory does not exist, (ii) Unix pathname.

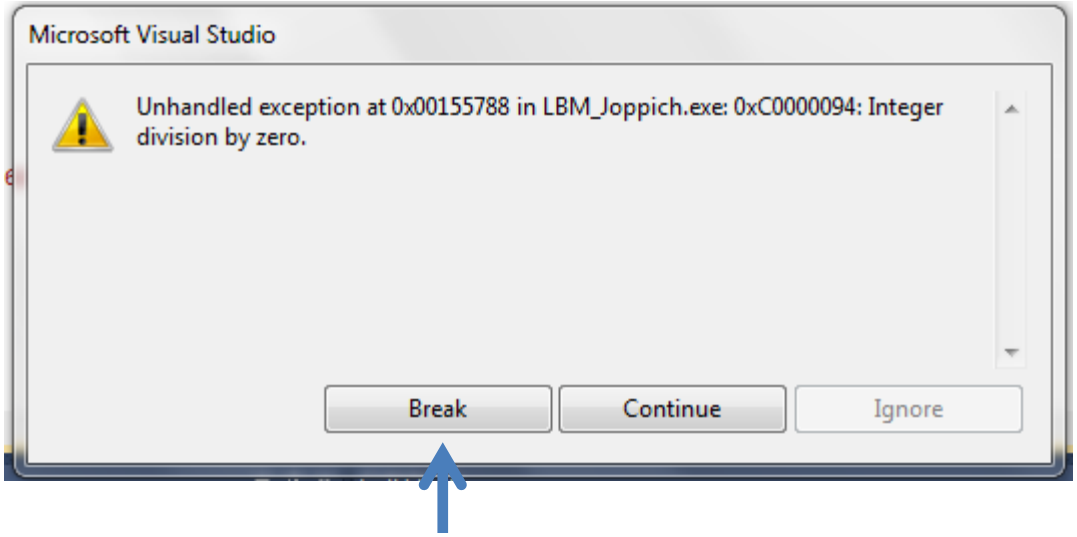
Getting it to work … (4/9)

- **Program changes in Step 1:**
  - Subdirectory LBMData2D created
  - In routine PrintInfo() pathnames changed to Windows-convention
  - In routine printGitter() pathnames changed to Windows-convention
  - In routine printFluid() pathnames changed to Windows-convention
- **Use Visual Studio to navigate the program source code:**



Getting it to work … (5/9)

- **Step 2: Again, run the program via Debug → Start Debugging. Within an unhandled exception occurs:**

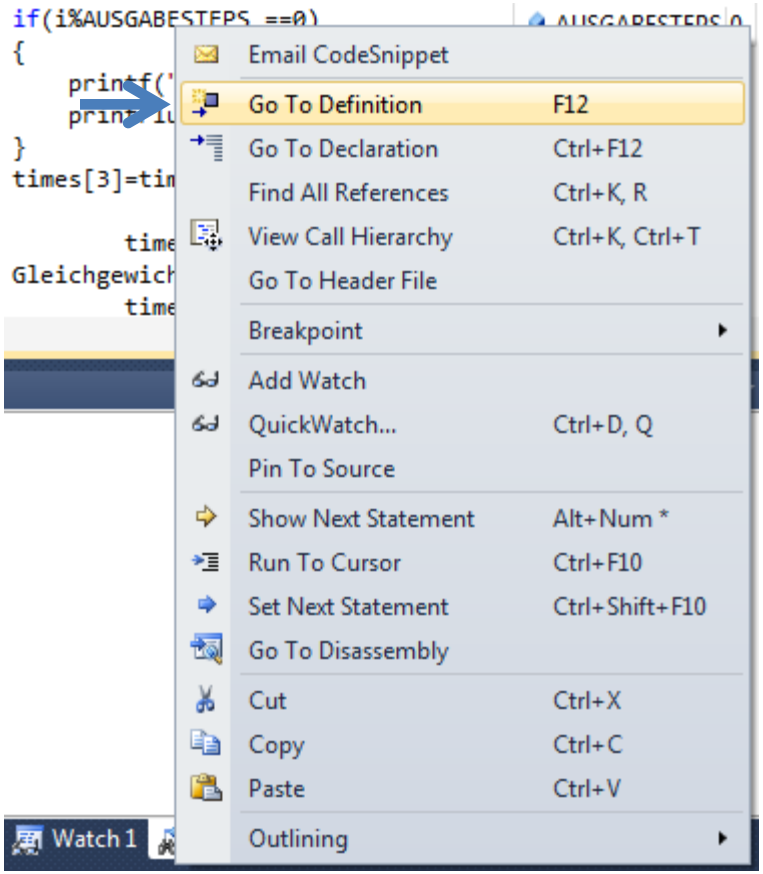


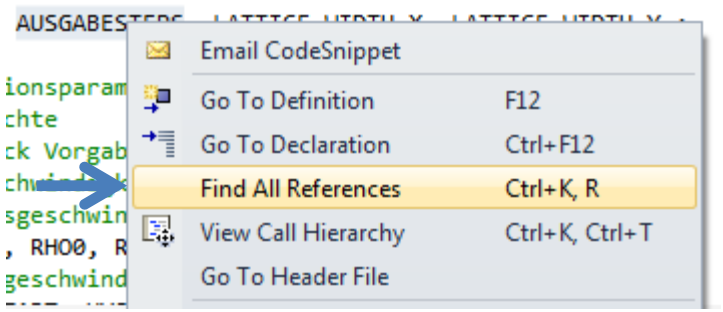
- **Variable AUSGABESTEPS has value 0:**

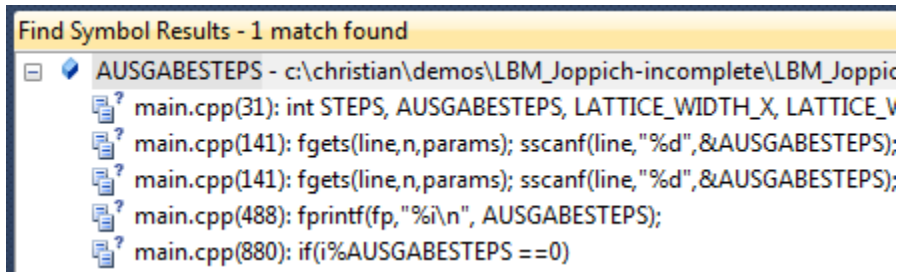
```
877         // Ausgabe
878
879                 times[3]=times[3]-clock();
880         if(i%AUSGABESTEPS ==0)                    AUSGABESTEPS 0
881         {
882             printf("\n Schritt %06d von %06d", i, STEPS);
883             printFluid(i);
884         }
```

Getting it to work … (6/9)

- **Use Visual Studio to navigate the program source code:**



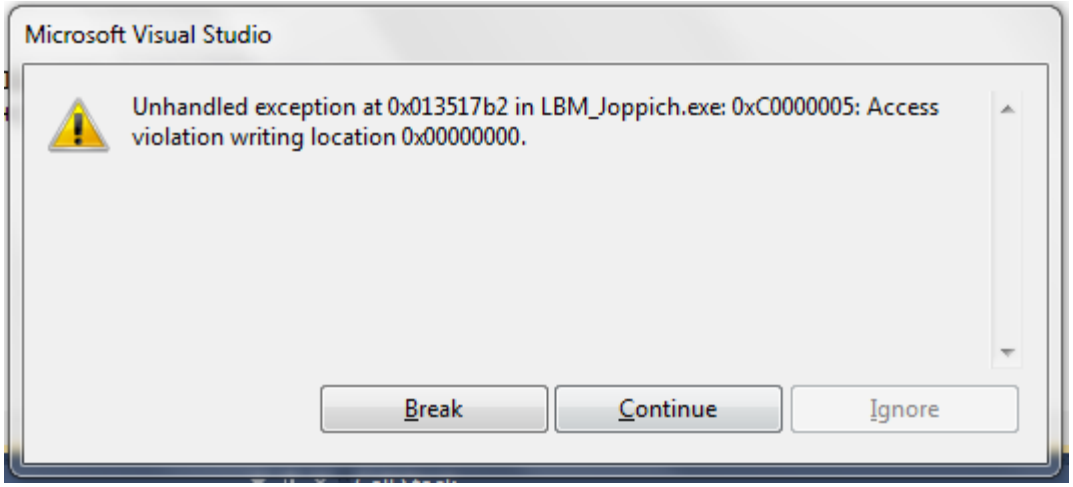




- Variable is read in function read_parametrs(). Problem: Input file wasn‘t found.

Getting it to work … (7/9)

- **Step 3: Again, run the program via Debug → Start Debugging. Programs now runs for some minutes, but then:**



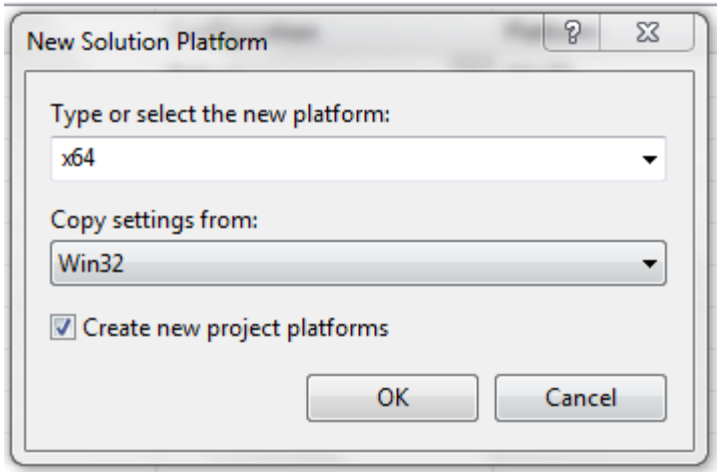
- **Location 0x00000000 is *strange*:**
  - Beginning of the Stack?
  - End of the Heap?
  - Uninitialized pointer?
  - Out of memory and pointer “overflow”?

Getting it to work … (8/9)

- **Looking at the Task Manager we see that the application‘s working set is about to exceed 2 GB, which is not possible on a 32bit OS:**

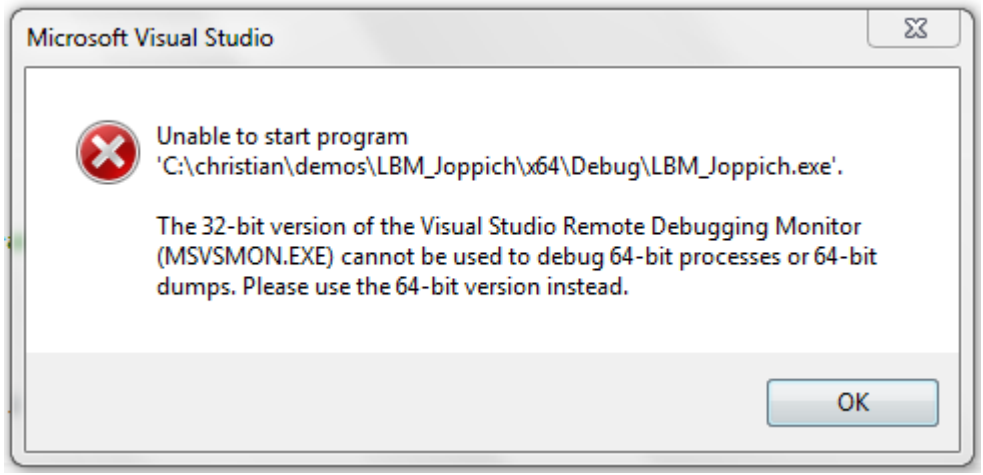
Abbildname	Benutzername	CPU	Arbeitsspeicher (privater Arbeitssatz)	Beschreibung
LBM_Joppich.exe	ct747764	00	2.074.008 K	LBM_Joppich.exe

- **Solution: Switch to 64bit. Build → Configuration Manager… and add a new platform. Make sure to copy the settings!**



Getting it to work … (9/9)

- **Step 4: Again, run the program via Debug → Start Debugging. The program does not run on 32bit system, but it does work fine on 64bit systems!**



  - If you change systems do not forget to adapt settings, i.e. working directory.

- **Step 5: Of course I always know what I am doing ;-), therefore I add /D_CRT_SECURE_NO_WARNINGS to get rid of these compiler messages…**

Performance Analysis (1/4)

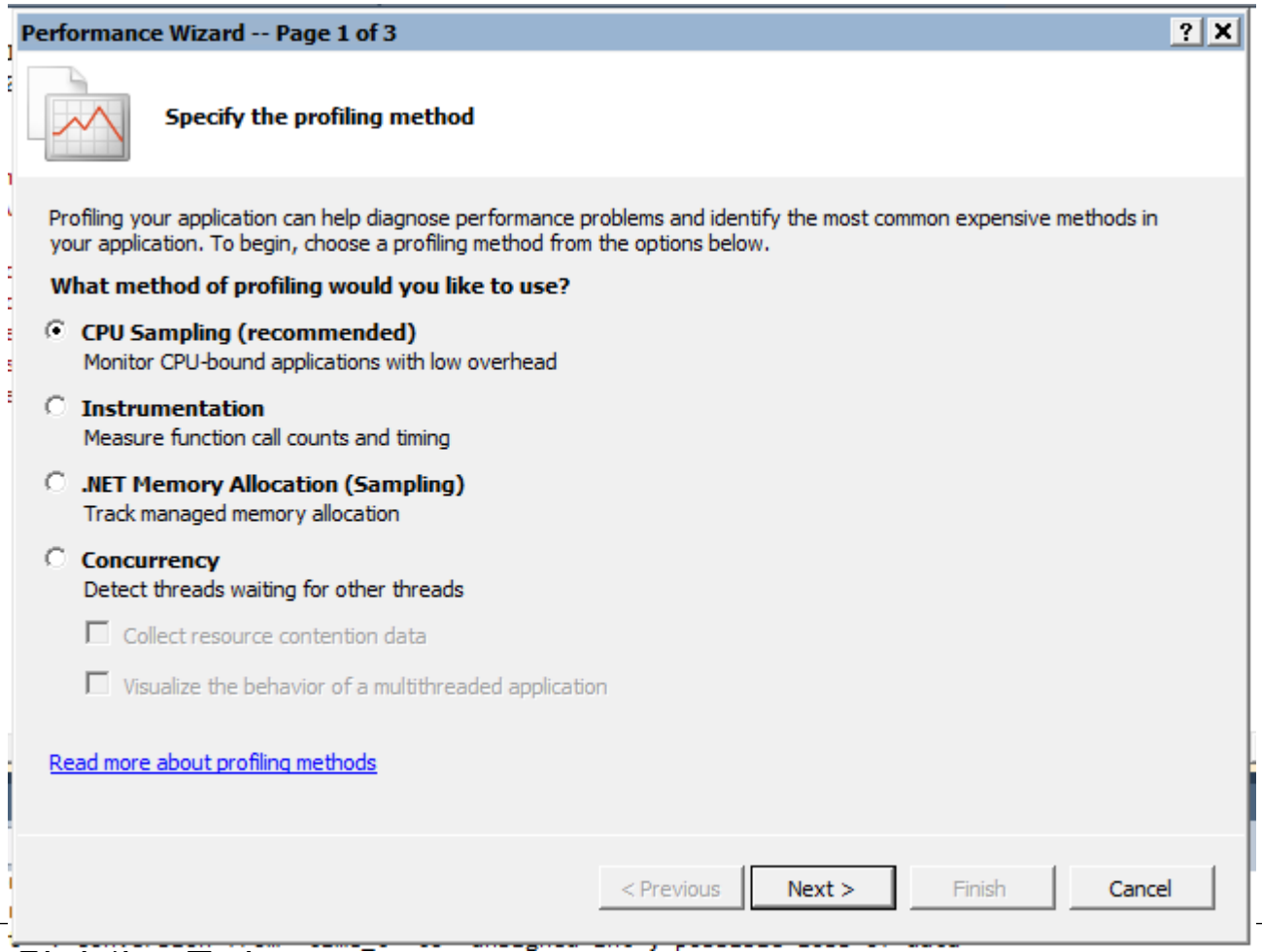
- **Step 6: Enabled OpenMP-aware compilation. Enabled optimization for SSE2 instructions. Simple performance analysis:**

Configuration	#Threads	Runtime (total time) as measured by the program [sec]
Debug	1	95.11
Release	1	30.17
Release (w/ SSE2)	1	31.61
Release	2	22.63
Release	4	24.20

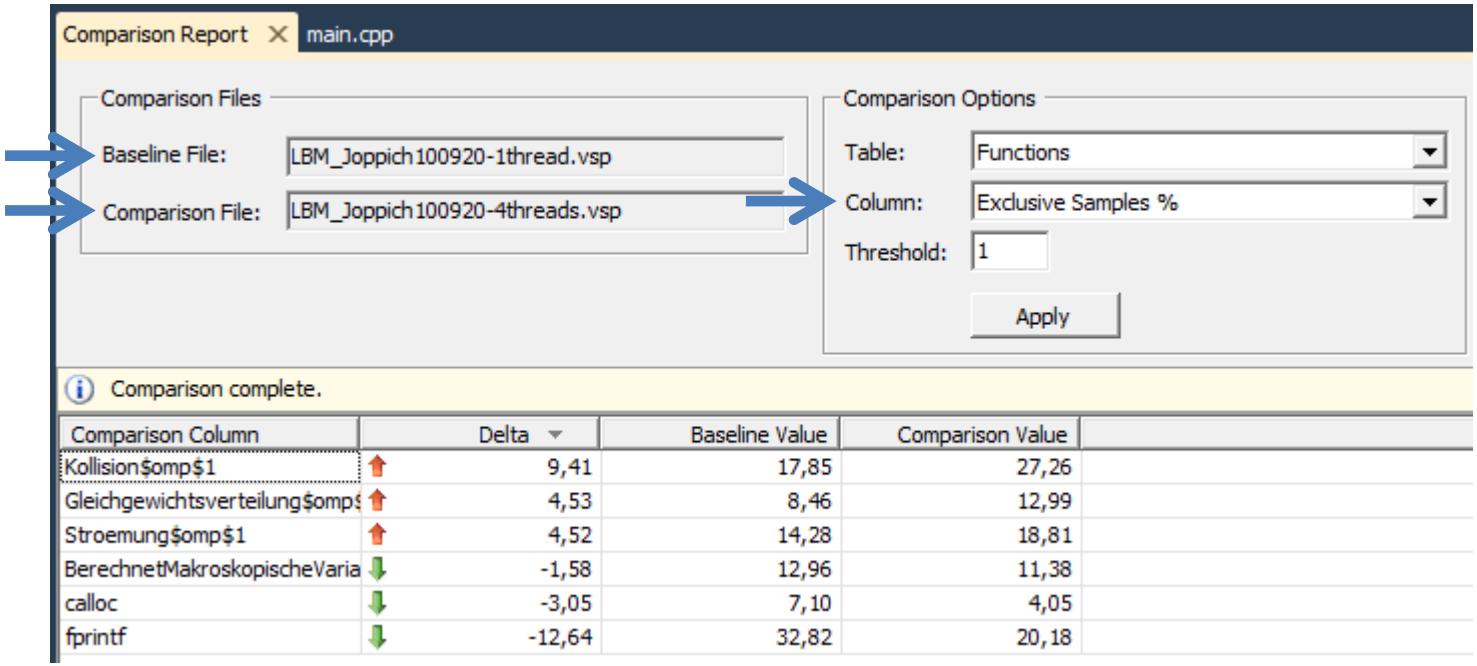
  - Compiler Optimizations help a lot
  - SEE2 support does not help, although it helps in many floating point-intense programs
  - OpenMP scalability is limited to about two threads

Performance Analysis (2/4)

- **Step 7: Do Sampling-Analysis of single-threaded and quad-threaded runs for comparison of runtime profile: Analyze → Launch Performance Wizard…**



Comparison:



Comparison Column	Delta	Baseline Value	Comparison Value
Kollisionomp1	9,41	17,85	27,26
Gleichgewichtsverteilungomp	4,53	8,46	12,99
Stroemungomp1	4,52	14,28	18,81
BerechnetMakroskopischeVaria	-1,58	12,96	11,38
calloc	-3,05	7,10	4,05
fprintf	-12,64	32,82	20,18

- Function‘s fraction in % of runtime changes
- Some functions get faster with four threads
- Some functions stay the same (or get slower) with four threads

Performance Analysis (4/4)

- **Step 8: Think! And maybe use advanced tools.**
  - Data distribution on cc-NUMA architectures?
  - Load Balancing?
  - Contention on Lock / Synchronization constructs?
  - …
- **Advanced Tools:**
  - Concurrency-Analysis of Visual Studio 2010
  - Intel Parallel Studio + Intel Threading Tools
- **… and in case of MPI programs:**
  - Intel Tracing Tools
  - TU Dresden‘s Vampir

The End.

- **Thank you for your attention.**