# Windows-HPC and Visual Studio Basics

*Christian Terboven*
*Center for Computing and Communication, RWTH Aachen University*
*Seffenter Weg 23, 52074 Aachen, Germany*
*terboven@rz.rwth-aachen.de*

## Abstract

This document guides you through the prepared examples and exercises. The purpose of the following tasks is to make you feel comfortable with the basic usage scenarios of our Windows-HPC environment and Visual Studio as a development environment, and to recapitulate the content of the demos presented during the talks. Please regard these tasks just as proposals of what you could do – if you prefer to work with your own code we happily support you as well.

For instructions on how to log in to the lab systems (laptops + desktop) and from there on how to access our Cluster during the Lab sessions please see the separate handout. Please remember that all tasks concerning the production environment and Visual Studio 2008 should be executed on `cluster-win.rz.rwth-aachen.de`, while all tasks regarding Visual Studio 2010 and Vampir have to be executed on `cluster-win-lab.rz.rwth-aachen.de`.

**If you need help or have any question please do not hesitate to ask!**

## 1. General Lab Preparation & File systems

**File systems.** Please login to the Windows-HPC Cluster, choose one of the frontend nodes described above. Click on Start → Computer in order to open the Explorer. You will find that our interactive nodes have two local hard drives, C: and D:. In addition, there are several network drives that are mounted by default. You will find at least:

- H: - This is your Home directory.
- W: - This is your Work directory.
- X: - This is your Documents directory.
- P: - This is a network drive containing several examples provided by the Computing Center.

**Task 1.1**: For each mounted network drive, find out how much space you have occupied and how much space is left. The total available space of the network drives is limited by your current quota that can be queried by using the `quota` tool on the command line.

| | Quota | Space occupied | Space left on device |
|---|---|---|---|
| H: | | | |
| W: | | | |
| X: | | | |

**Lab Preparation.** The network drive P: is read-only. In order to work on the exercise you have to copy the directory `P:\PPCES_2010\` including all subdirectories to your Home directory.

**Task 1.2**: Copy the lab exercises to your Home directory.

## 2. Batch System

**Batch System.** While the interactive machines are intended for software development tasks and small test runs, you should use the batch system for resource-intensive and / or long-running compute jobs. There are two possible ways for a user to submit jobs to the batch system: Either via the GUI, or via the command line. In order to avoid waiting times we have created a node group named RZ_HPCLAB that should be used to submit jobs to during the lab sessions.

You will find an executable named `mpi_HelloWorld_vs2005` in the directory `P:\PPCES_2010` (you already should have copied it to your Home directory `H:\`). This program will be used to test some aspects of batch job submission during the following two tasks. Please remember that a batch job cannot access network drives via the drive letter, so please use `\\cifs\cluster\home\<userid>` instead of `H:`.

**Task 2.1**: Job Submission via the GUI.

- Start the HPC Job Manager via Start → All Programs → Microsoft HPC Pack → HPC Job Manager. Familiarize yourself with the Job Management pane on the left hand side of the program window. It can be used to restrict the view of the total list of jobs in the cluster to a subset you are interested in.
- Submit the `mpi_HelloWorld_vs2005` program for execution on (i) two cores, (ii) two sockets and (iii) two nodes by creating jobs in the HPC Job Manager GUI. Specify a file to contain the program's output. After the jobs have been completed, examine the output files to find out the names of the compute nodes the program has been executed on.
- Use the GUI to find out on which nodes your jobs have been executed.

**Task 2.2**: Job Submission via the Command Line Interface.

- Submit the `mpi_HelloWorld_vs2005` program to be executed on (i) two cores, (ii) two sockets and (iii) two nodes by creating jobs using the `job` command. You can use the `submit_hpc2008.cmd` file as a template, but you have to change it to use the location you copied the files to. Specify a file to contain the program's output. After the jobs have been completed, examine the output files to find out on which nodes the program has been executed.
- Use the `job` command to find out on which nodes your jobs have been executed.

## 3. Getting Help

If you need help with our batch system, there is a more detailed introduction at our Website http://www.rz.rwth-aachen.de/go/id/pux/. You may also find help in our HPC-Primer which can be found online at http://www.rz.rwth-aachen.de/go/id/pil/. For help in Visual Studio press F1 and you will get to the Online Help. A lot of questions regarding Windows programming can also be solved on http://www.msdn.com. Of course you can always ask us if you have any question, or send an email to support@rz.rwth-aachen.de.

## 4. First steps with Visual Studio

Some of the tools covered in the labs depend on Visual Studio 2008, such as tasks using the Intel Parallel Studio and the DDTlite MPI debugger. Some other tasks explicitly make use of the new features only available in Visual Studio 2010. The exercises on general parallel programming can be used with both Visual Studio versions.

**Task 4.1**: Create a Visual Studio Solution for the Matrix-Vector-Multiplication code. The source code is provided in the directory `P:\PPCES_2010\vs_basics\src`. If you want to skip the manual creation of a Visual Studio Solution just proceed to Task 4.2.

**Task 4.2**: This task is about comparing and tuning the performance of a Matrix-Vector-Multiplication code. The source code along with a Visual Studio Solution can be found in directory `P:\PPCES_2010\vs_basics\src_with_vsproject`. If you open this solution with Visual Studio 2010 you have to follow the conversion wizard. If you have created your own Visual Studio Solution in task 4.1 and it is working fine, you can use your own solution during the course of this task.

- Build the solution and execute it. Hint: Use *Debug* → *Start without Debugging* in order to not have the debugger influence the performance and get the chance to view the program's output. The performance of five different versions of the Matrix-Vector-Multiplication routines is printed as the program's output. Can you explain the performance differences?
    - MxV Row: Row-wise access to the data structures.
      ```
      for (i = 0; i < m; i++) {
              a[i] = 0.0;
              for (j = 0; j < n; j++)
                      a[i] += b[i*n + j] * c[j];
      }
      ```
    - MxV Col: Column-wise access to the data structures.
      ```
      for (i = 0; i < m; i++)
              a[i] = b[i*n] * c[0];
      for (j = 1; j < n; j++)
              for (i = 0; i < m; i++)
                      a[i] += b[i*n + j] * c[j];
      ```
    - MxV [Row | Col] OMP: In the Advanced lab exercises throughout the next couple of days you will parallelize these routines with OpenMP. As of now, they are the same as the routines discussed above.
    - MxV MKL: In the last part of this task you will be asked to use the Intel Math Kernel Library (MKL) to perform the operation.
- Compare the performance of the routines with both the Debug and the Release compiler Configuration. Furthermore, increase the optimization level of the Microsoft compiler by enabling selected optimizations. How far can you drive the code's performance?

**Task 4.3**: (VS2008 only) You might want to try the Intel Compiler by right-clicking on the Solution and selecting Intel *C++ Compiler Pro* → *Use Intel C++*. Increase the optimization level of this compiler as well. How far can you drive the code' performance?

**Task 4.4**: (VS2008 only) Change the `mxv_mkl()` routine such that it calls `cblas_dgemv` from the Intel MKL (look in *Start* → *All Programs* → *Intel Software Development Tools* menu for help). In order to use the MKL on our system, you have to add `C:\Program Files\Intel\MKL\10.1.3.028\include` as an Include Path and `C:\Program Files\Intel\MKL\10.1.3.028\ia32\lib` (32-bit) or `C:\Program Files\Intel\MKL\10.1.3.028\em64t\lib` (64-bit) as a Library Path and add `mkl_intel_c.lib mkl_intel_thread.lib mkl_core.lib libiomp5md.lib` (32-bit) or `mkl_intel_lp64.lib mkl_intel_thread.lib mkl_core.lib libiomp5md.lib` (64-bit) as an additional library to link with. Which performance (MFLOPS rate) can you achieve?

**Task 4.5**: (VS2010 only) Use the *CPU Sampling* Profiler to examine how much time (in seconds) is spent in every individual version of the MxV implementations. Visual Studio 2010 will ask for elevated privileges to perform the sampling run. How can the MFLOPS rate be derived from the runtime measured in seconds?

## 5. Debugging with Visual Studio

The following tasks are intended to make you familiar with the Visual Studio debugger for MPI and OpenMP programs. Please decide whether to use Visual Studio 2008 or Visual Studio 2010. Please do not despair of the debugger immediately and try to follow the instructions given during the presentation. After some time of experimenting you will get a *feeling* for the parallel debugger – in the end of the lab the solution to the exercises will be presented live.

**Task 5.1**: Preparing the MPI-Debugger. Go to the `jacobi_SendRecv.vs2008` or `jacobi_SendRecv.vs2010` directory and open the `jacobi_SendRecv_win.sln` solution by double-clicking, Visual Studio 2008 or Visual Studio 2010 will start. By examining the Solution Explorer you will find that it consists of three projects, namely the Jacobi solver written in C and C++ (both using the Microsoft compiler) and Fortran (using the Intel compiler) – the latter is not included in the Visual Studio 2010 solution. You should decide for one language by selecting the project, right-clicking and choosing *Set as StartUp project*. Compile the project using the *Debug* configuration. **Note:** Visual Studio 2008 only allows one MPI debugging session per machine – please coordinate the following task with the lab instructor. Visual Studio 2010 allows for many simultaneous debugging sessions.

**Task 5.2**: Performing the MPI-Debugging. You may find the *Processes* window in the menu under *Debug* → *Windows* → *Processes*. Execute the program with 2 MPI processes. Your task is to examine the value of variable `fLRes` for `data.IterCount == 2`, `j == 1000` und `i == 2`. The result is: -*1.2012004196393182e-006*.

**Task 5.3**: Preparing the OpenMP-Debugger. You continue to work in the same directory with the same project. In order to ease the debugging I recommend to only use 1 MPI process. Enable the recognition of OpenMP pragmas.

**Task 5.4**: Performing the OpenMP-debugging. You may find the *Threads* window in the menu under *Debug* → *Windows* → *Threads*. Execute the program 2 OpenMP threads. Your task is to examine the value of variable `fLRes` for `data.IterCount == 2`, `j == 2` und `i == 2`. The result is: *1.9616983724916493e+065*.

## 6. Performance Optimization of Hybrid (= MPI + OpenMP) Programs

In the following tasks you can try yourself with a hybrid parallel program that is using both MPI and OpenMP. The goal is to optimize the performance of the program by selecting the right number of MPI processes and OpenMP threads per node. We recommend using the batch system for the performance measurements.

**Task 6.1**: Examining the Hybrid Parallelization. Select either the `jacobi_SendRecv.vs2008` or `jacobi_SendRecv.vs2010` directory and open the `jacobi_SendRecv_win.sln` solution by double-clicking, Visual Studio 2008 or Visual Studio 2010 will start. By examining the Solution Explorer you will find that it consists of three projects, namely the Jacobi solver written in C and C++ (both using the Microsoft compiler) and Fortran (using the Intel compiler) – the latter is not included in the Visual Studio 2010 solution. You should decide for one language by selecting the project, right-clicking and choosing *Set as StartUp project*. Make sure that OpenMP support has been enabled in the project as well.

**Task 6.2**: Finding the optimal job configuration. The RZ_HPCLAB nodegroup consists of two nodes, you may want to use the *HPC Job Manager* to find out more details. Try to find the settings (= number of MPI processes and OpenMP threads per process) that delivers the best performance, but submitting an array of batch jobs. Which MFLOPS rate do you get?