Compute Cluster Server Lab 2: Carrying out Jobs under Microsoft Compute Cluster Server 2003

Compute Cluster Server Lab 2: Carrying out Jobs under Microsoft Compute Cluster Server 20031
Lab Objective 1
General Scheme of Carrying out the Jobs under Microsoft Compute Cluster Server 2003 1
Exercise 1 – Compiling a Program for Running under CCS 2003 3
Task 1 – Installation of Microsoft Compute Cluster Pack SDK.....3
Task 2 – Setting the Development Integration Environment of Microsoft Visual Studio 2005 6
Task 3 – Compiling a Parallel Program in Microsoft Visual Studio 2005.....9
Exercise 2 – Running a Serial Task 14
Launching the Program via Graphic User Interface.....14
Launching the Program by Means of Template.....23
Launching the Program from the Command Line.....27
Exercise 3 – Launching a Parallel Job 29
Exercise 4 – Launching a Parametric Sweep..... 36
Exercise 5 – Launching a Work Flow 43
Optional Exercise. Evaluating the Network Performance Parameters..... 52
General Network Performance Parameters.....52
Methods for Evaluating the Network Performance Parameters.....53
Compiling the Benchmark Program.....53
Running the Benchmarks.....53
Discussions..... 58

In order to use the high performance cluster effectively, it is necessary to use a family of quite complicated software systems. For a long time the users of Windows clusters have to use simultaneously the software from several vendors. With the release of Compute Cluster Server 2003 (CCS) Microsoft company provided a full spectrum of software, which is necessary for the efficient use of clusters and the development of the parallel programs, which fully use the available computational power.

Lab Objective

The lab objective is to learn how to compile and start programs under the management of Microsoft Compute Cluster Server 2003. The lab assignments include:

- Exercise 1 – Compile a program to be run under CCS 2003.
- Exercise 2 – Launch a serial job
- Exercise 3 – Launch a parallel job
- Exercise 4 – Launch a parametric sweep
- Exercise 5 – Launch a work flow

Estimated time to complete this lab: **90 minutes**.

General Scheme of Carrying out the Jobs under Microsoft Compute Cluster Server 2003

To use the computational cluster resources efficiently, it is necessary to provide not only the immediate mechanisms of starting the jobs to be executed, but also to provide the management environment, which should manage the course of executing the jobs and solve the problem of efficient resource distribution. These tasks are efficiently solved with the help of the means embedded to CCS 2003.

Let us define the most important concepts used in CCS 2003:

- **A job** is a request for the allocation of the cluster computational resources for carrying out a task. Each job may consist of one or more tasks,
- **A task** is a command or a program (including the parallel ones), which has to be executed on the cluster. A task cannot exist outside a certain job. A job may contain one or more tasks,

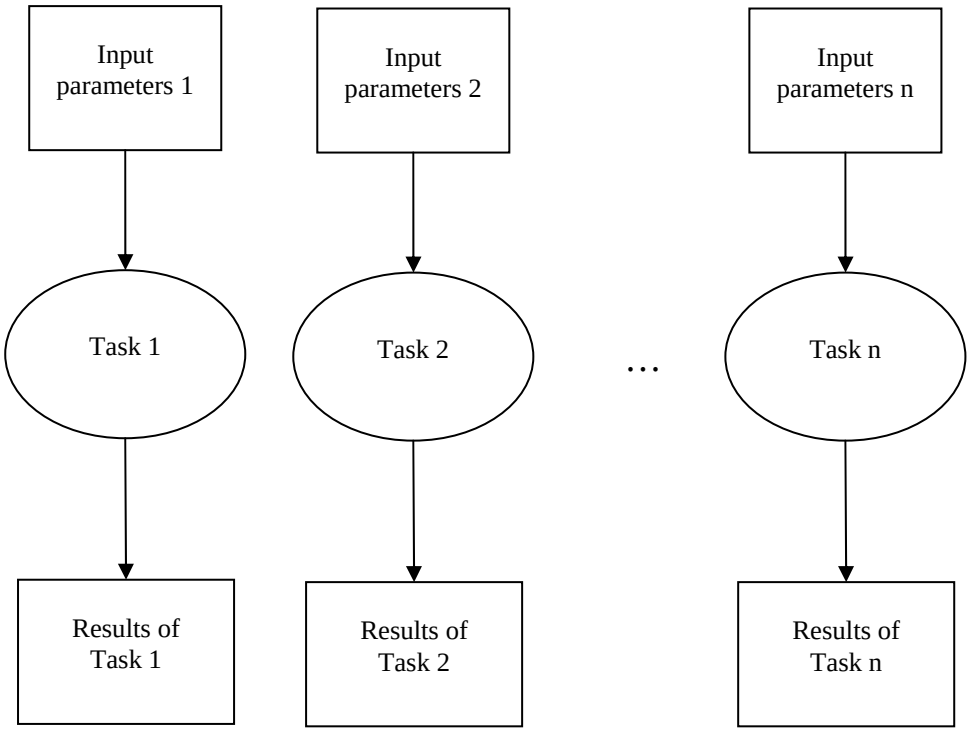
- **A job scheduler** is a service providing the job queue, allocating system resources, queuing the tasks and controlling the state of the executed tasks,
- **A node** is a computer, which is included into a cluster controlled by CCS 2003,
- **A processor** is one of the node computational devices,
- A **queue** is the list of tasks to be executed on the cluster, which are sent to the job scheduler. The order, in which the tasks are executed on the cluster, is determined by the planning policy adopted on the cluster,
- **A task list** is an equivalent of the job queue for the tasks of each concrete job. The order of executing the tasks will be determined by the FCFS strategy (first come, first served), if the user has not purposefully set some other order.

A job scheduler CCS 2003 operates both the serial and parallel tasks. The task is referred to as serial if it uses the resources of only one processor. The task is regarded to be parallel if it consists of several processes (or threads), which interact with each other in order to solve the task. As a rule, parallel MPI tasks require several processors for efficient execution. If MPI is used to provide message transmissions between parallel processes, then the parallel program may be executed on different cluster nodes. CCS 2003 includes its own realization of the standard MPI2: the library Microsoft MPI (MS MPI). If MS MPI is used, it is necessary to run parallel tasks using the special utility **mpiexec.exe**, which provides a simultaneous start of several parallel program copies on the selected cluster nodes. It should be noted that the immediate task start is the responsibility of the job scheduler, and the user can only add a program to the queue as its starting time is chosen automatically by the system depending on the availability of the computational resources and the tasks waiting in the queue for allocating the resources. Thus, it is necessary to perform the following operations to execute a program under CCS 2003:

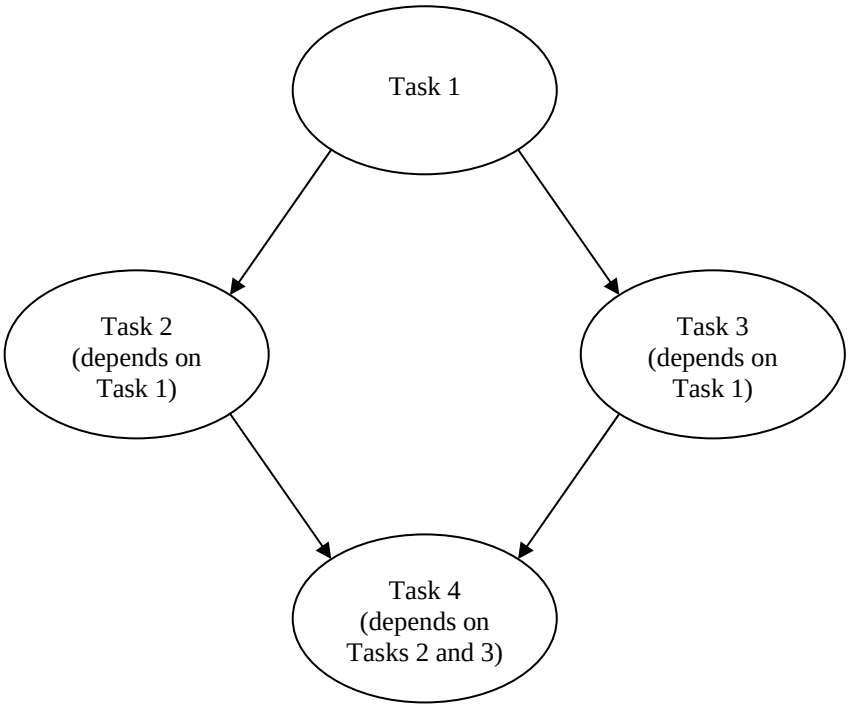
- To create a job describing the computational resources necessary for its execution,
- To create a task. The task is defined by means of a command. The execution of the command leads to running serial or parallel programs on the cluster. For instance, a parallel task is described by means of the command **mpiexec.exe** with the corresponding parameters (the list of the nodes for its executing, the name of the parallel program, the arguments of the command program line etc.),
- To add the task to the job created previously.

There are two special types of jobs:

- **A parametric sweep** is one and the same program (serial or parallel), several copies of the program are executed (possibly simultaneously) with various input parameters and various output files,



- **A work flow** is the case when several tasks (possibly the same program with different input parameters) are executed in a certain sequence. The sequence of execution is determined, for instance, by the task dependence against the computational results of the other tasks.



It will be demonstrated further in the lab, how to compile and run serial and parallel tasks under CCS 2003. Besides, there will be given examples of the parametric sweep and the work flow.

Exercise 1 – Compiling a Program for Running under CCS 2003

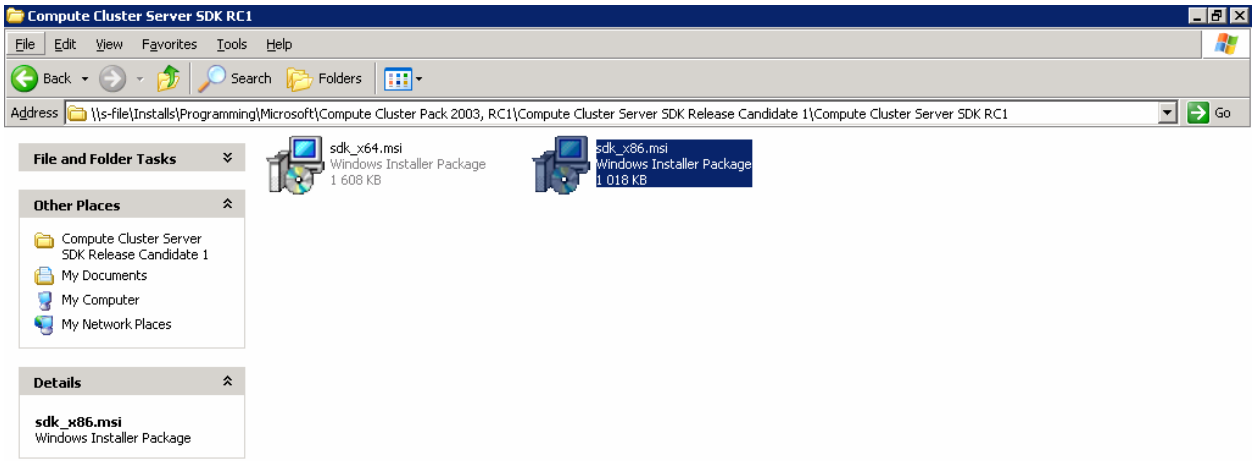
As it has been stated preciously, Microsoft Compute Cluster Server 2003 makes possible to control the execution of serial and parallel tasks. The parallel MPI tasks can be built with any MPI implementation (though the MS MPI implementation is preferable). Besides, it is possible to use other technologies for supporting the parallel programming (for instance, programming with the use of OpenMP).

This part describes only compiling parallel programs for MS MPI in Microsoft Visual Studio 2005.

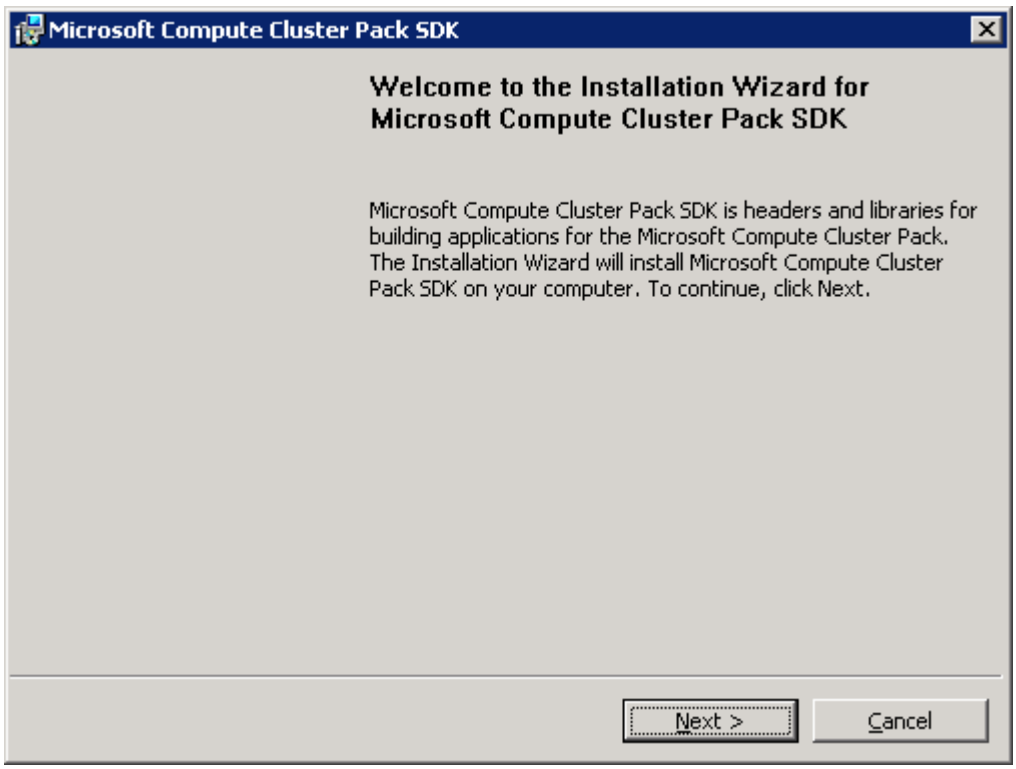
Task 1 – Installation of Microsoft Compute Cluster Pack SDK

In order to compile parallel programs operating in the environment MS MPI, it is necessary to install **SDK (Software Development Kit),** which is the set of interfaces and libraries for calling MPI functions:

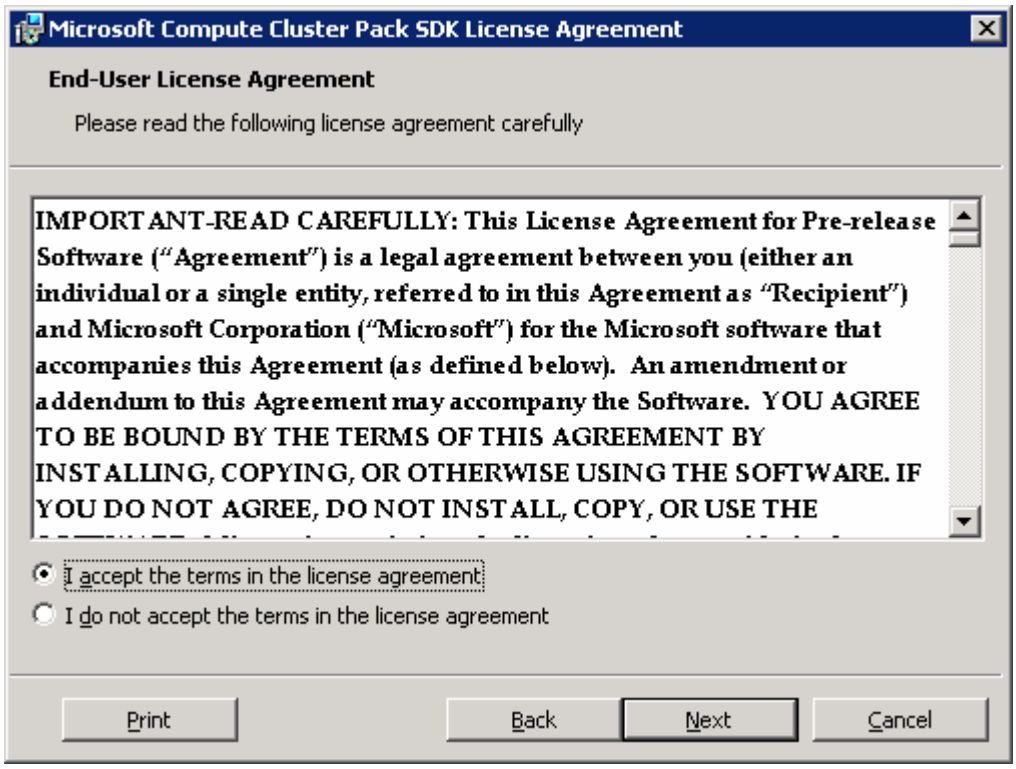
- Open the directory containing the download version of SDK (the description of the download procedure may be found in the Compute Cluster Server Lab 1 “Installation of Microsoft Compute Cluster Server 2003”) and run the installation program, which corresponds to your processor (32 bit or 64 bit version),



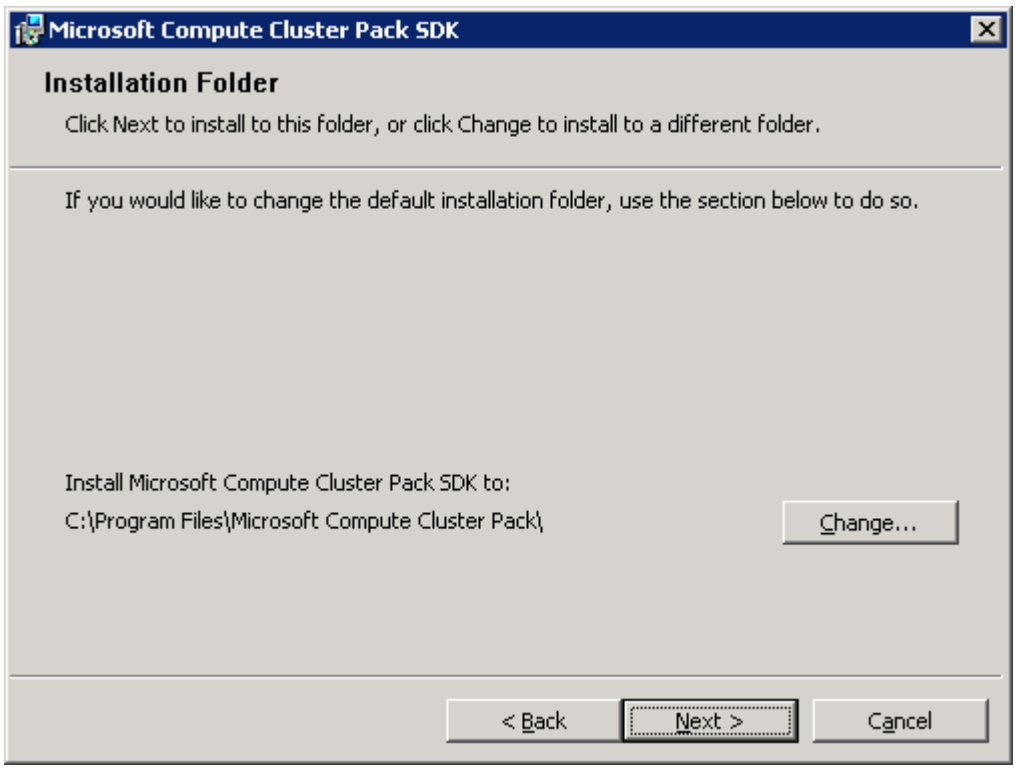
- In the opened window press the button **Next** to start the installation process,



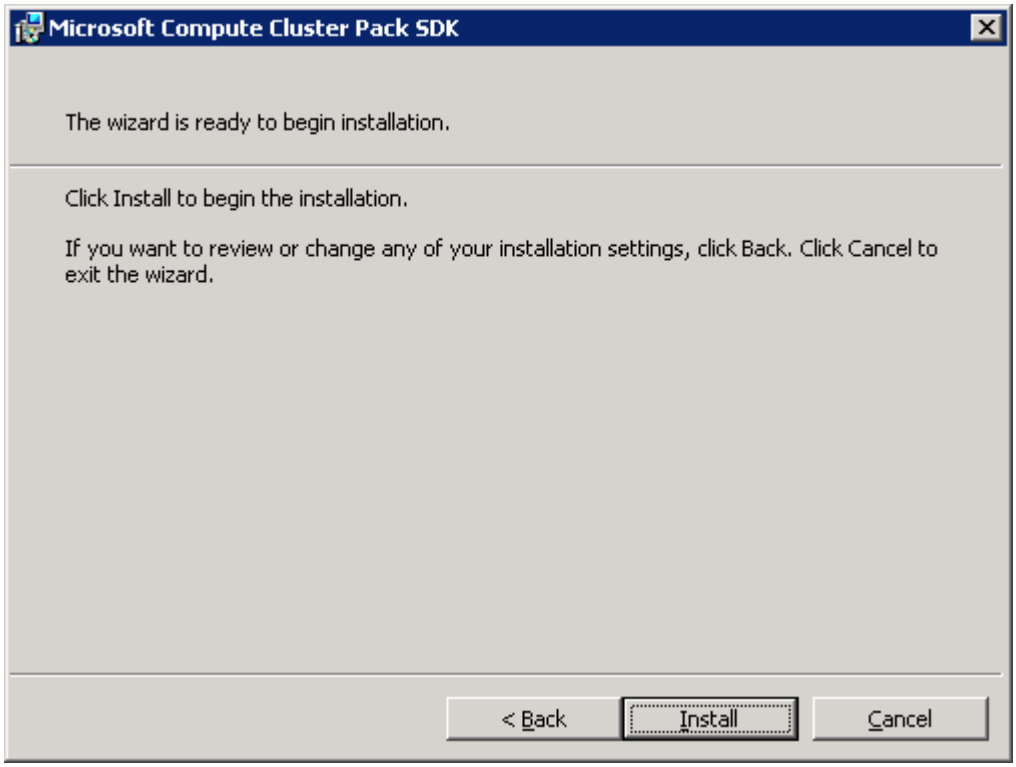
- Read the license agreement carefully. Choose the option "**I accept the terms in the license agreement**" in case you agree to the license agreement terms of using the system CCS 2003 and press the button **Next**,



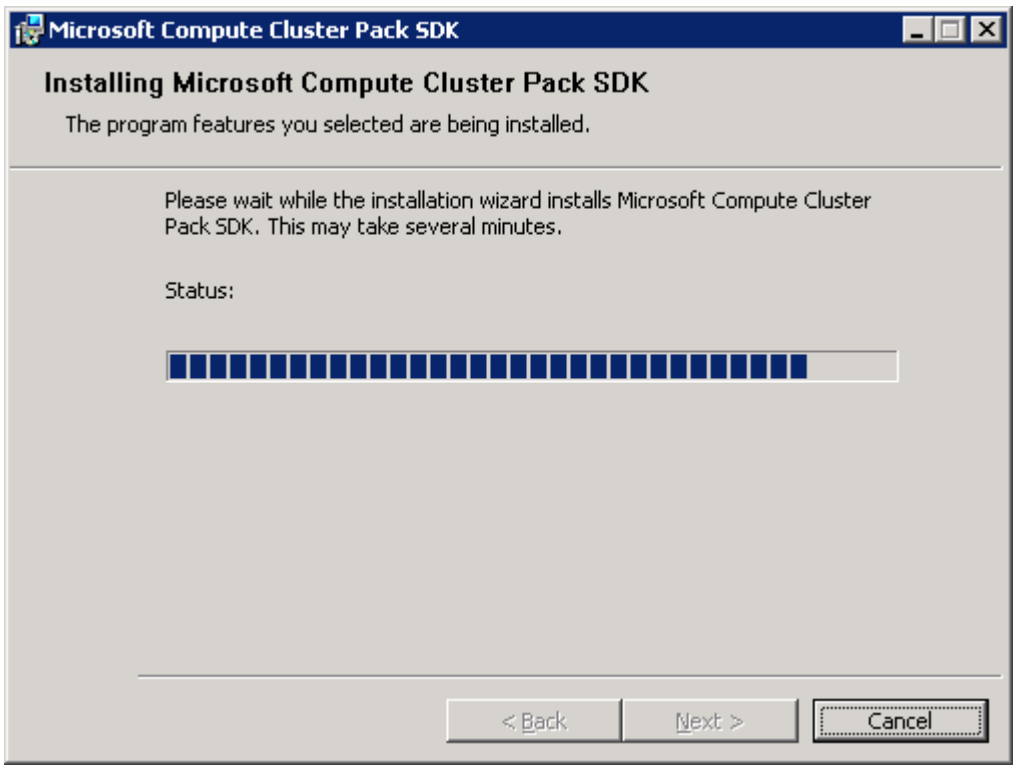
- Choose the directory where you are going to install SDK. To change the standard directory, press the button **Change**. Press the button **Next**,



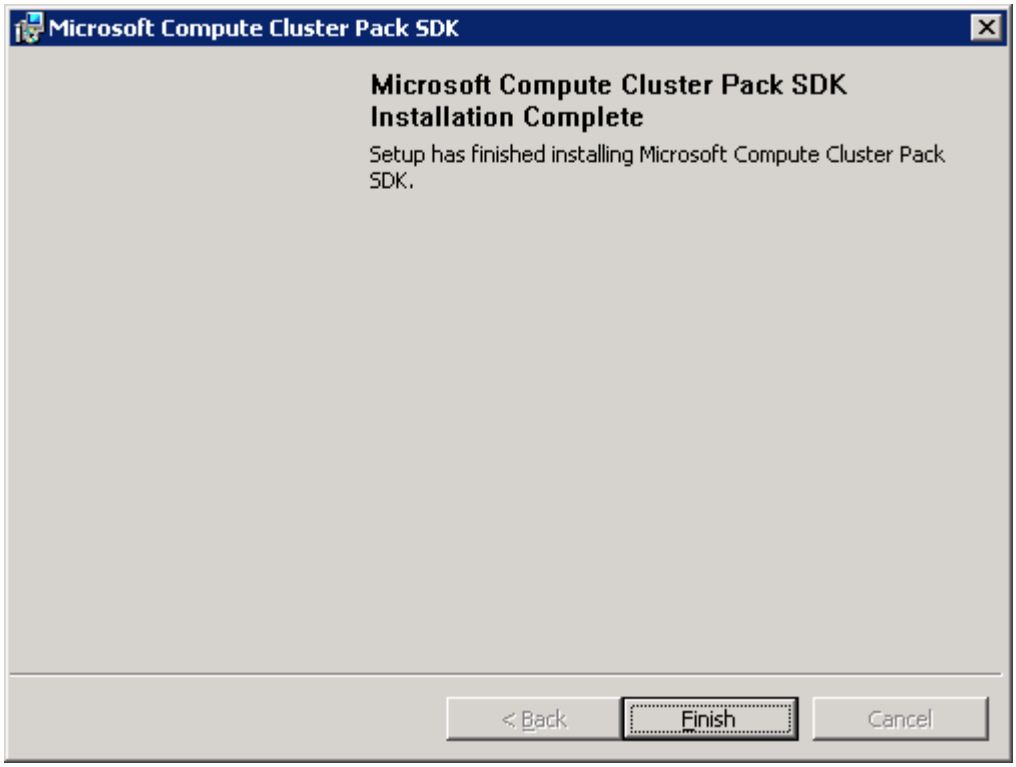
- In the new window press the button **Install** to start the installation of SDK,



- Wait till the program of SDK installation copies the required files,



- After copying the necessary files, press the button **Finish**,

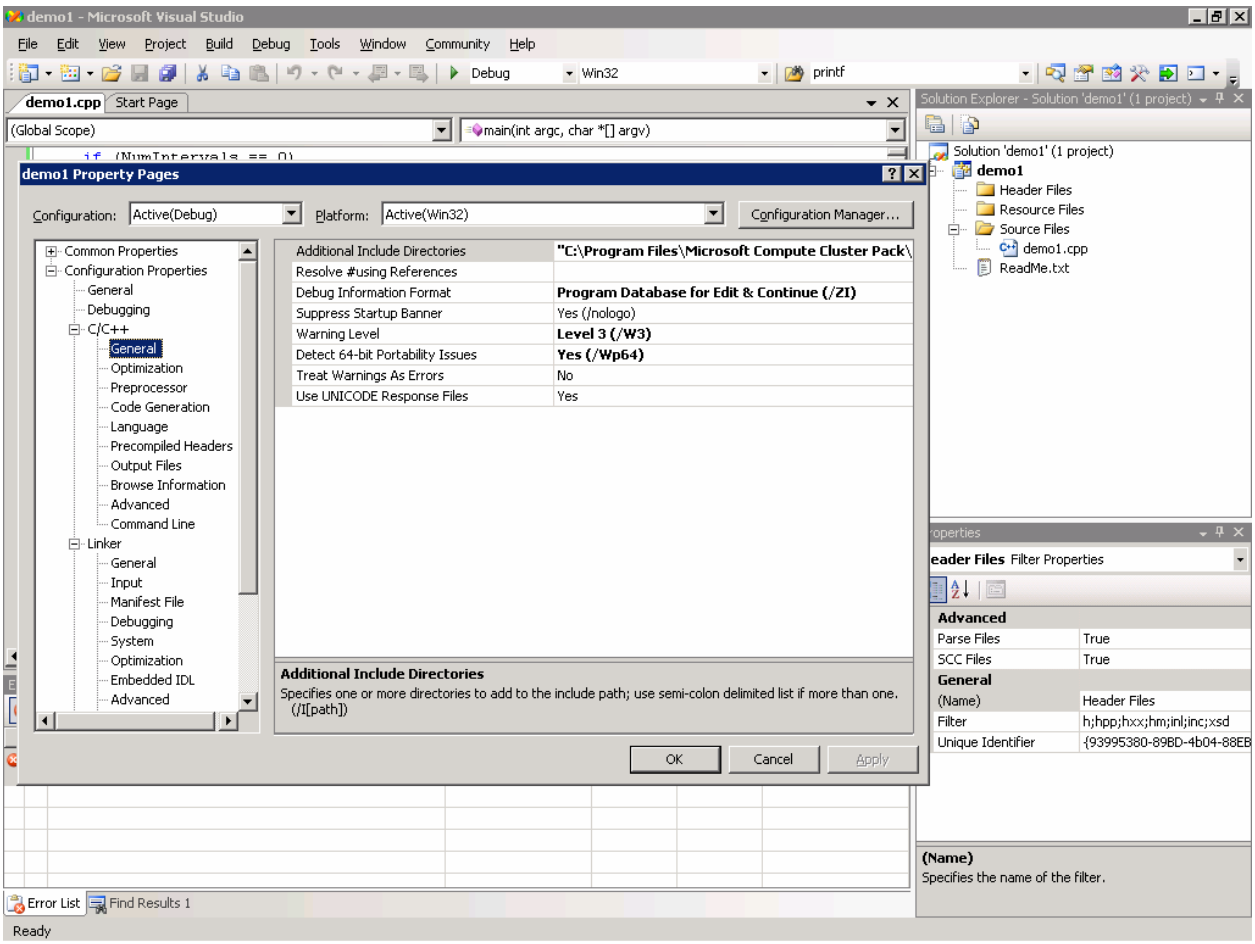


- Congratulations! The installation of Microsoft Compute Cluster Server 2003 SDK is completed.

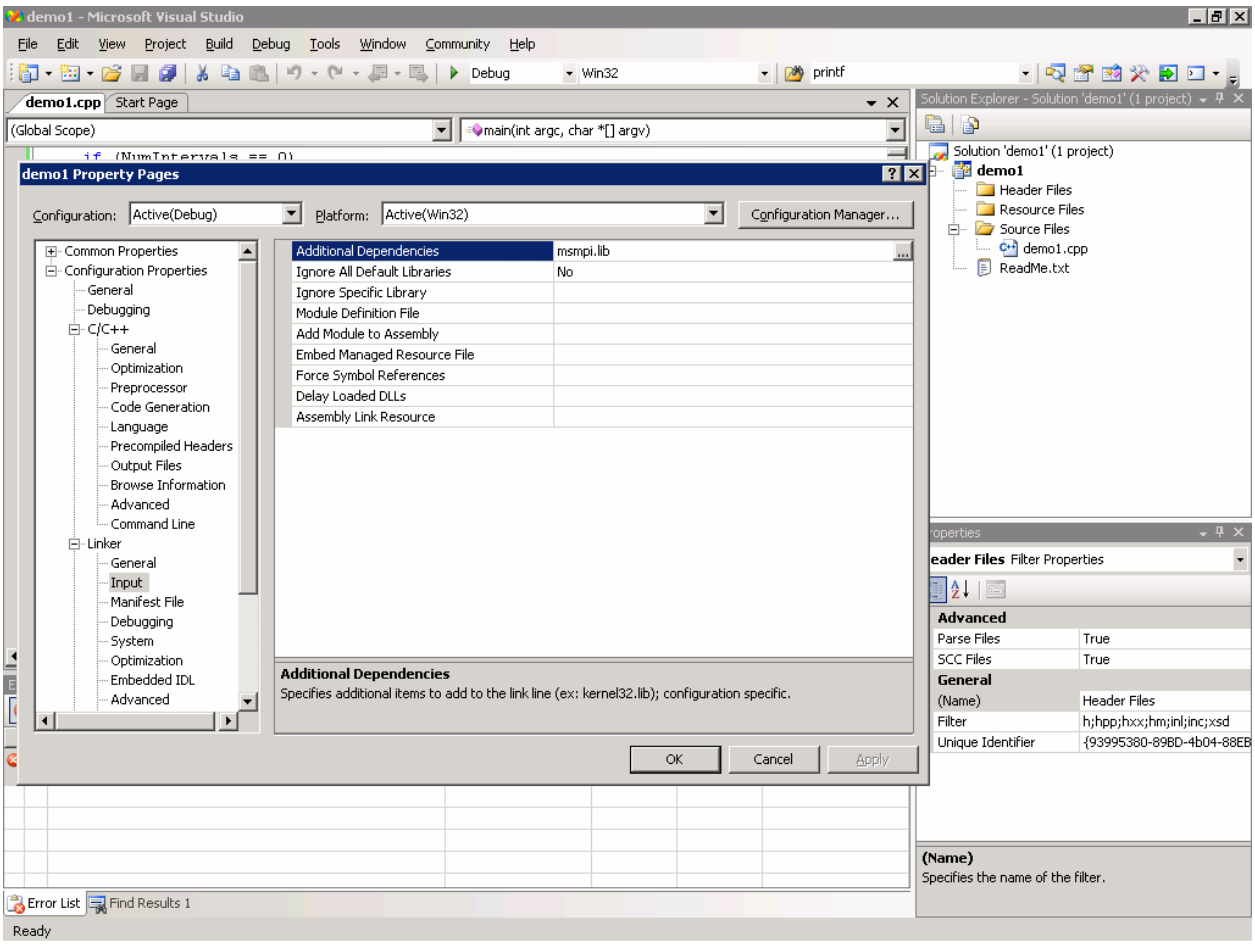
Task 2 – Setting the Development Integration Environment of Microsoft Visual Studio 2005

In order to compile the program using MS MPI, it is necessary to change the following project settings on default in Microsoft Visual Studio 2005:

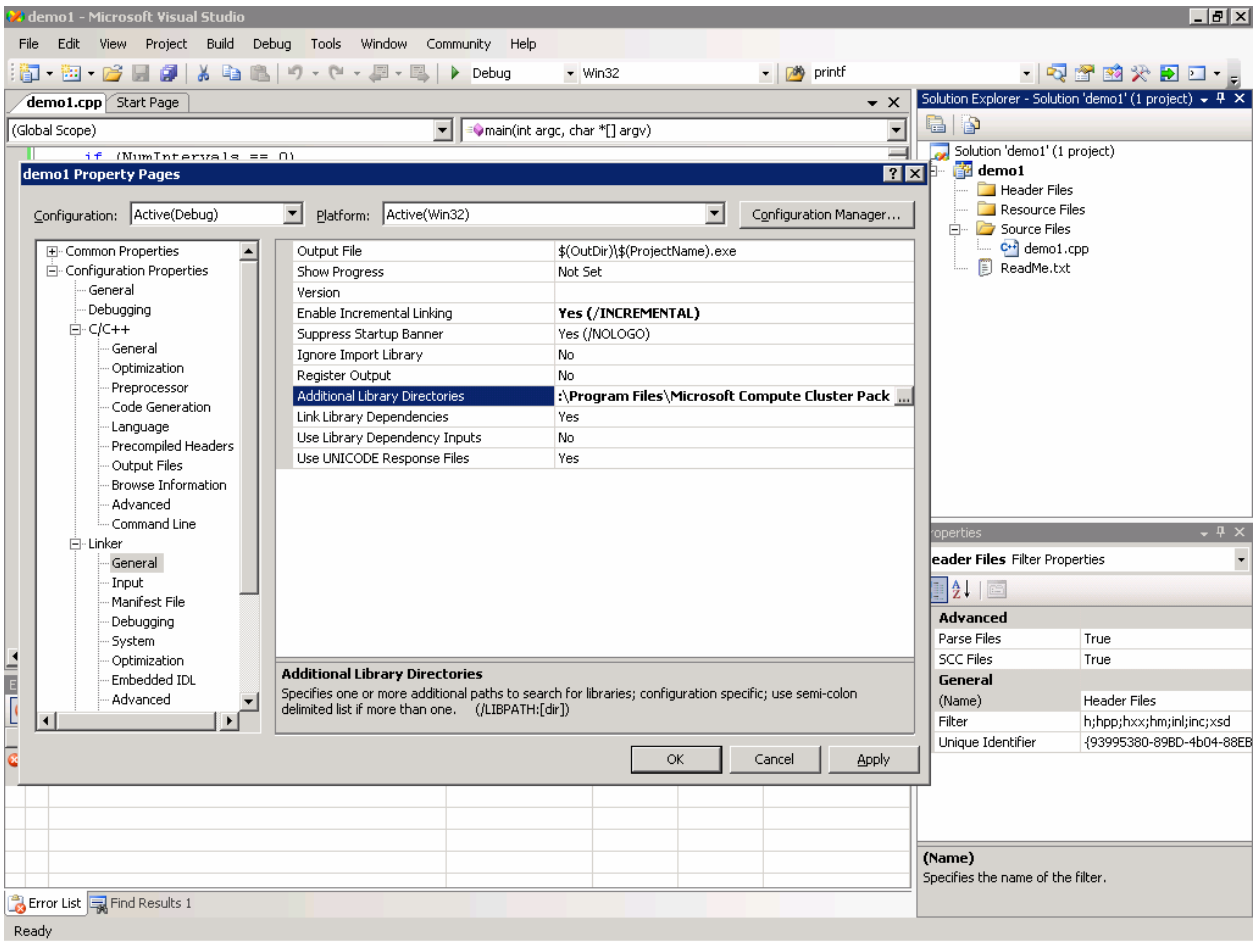
- **The path to the header files of MPI declaration**. Choose the menu option **Project->Project Properties**. In the option **Configuration Properties->C++->General->Additional Include Directories** enter the path to the header files of MS MPI: **<Installation Directory CCS SDK>\Include,**



- **The library file with the realization of MPI functions**. Choose the menu option **Project->Project Properties**. In the option **Configuration Properties->C++->Linker->Input->Additional Dependencies** enter the name of the library file **msmpi.lib,**



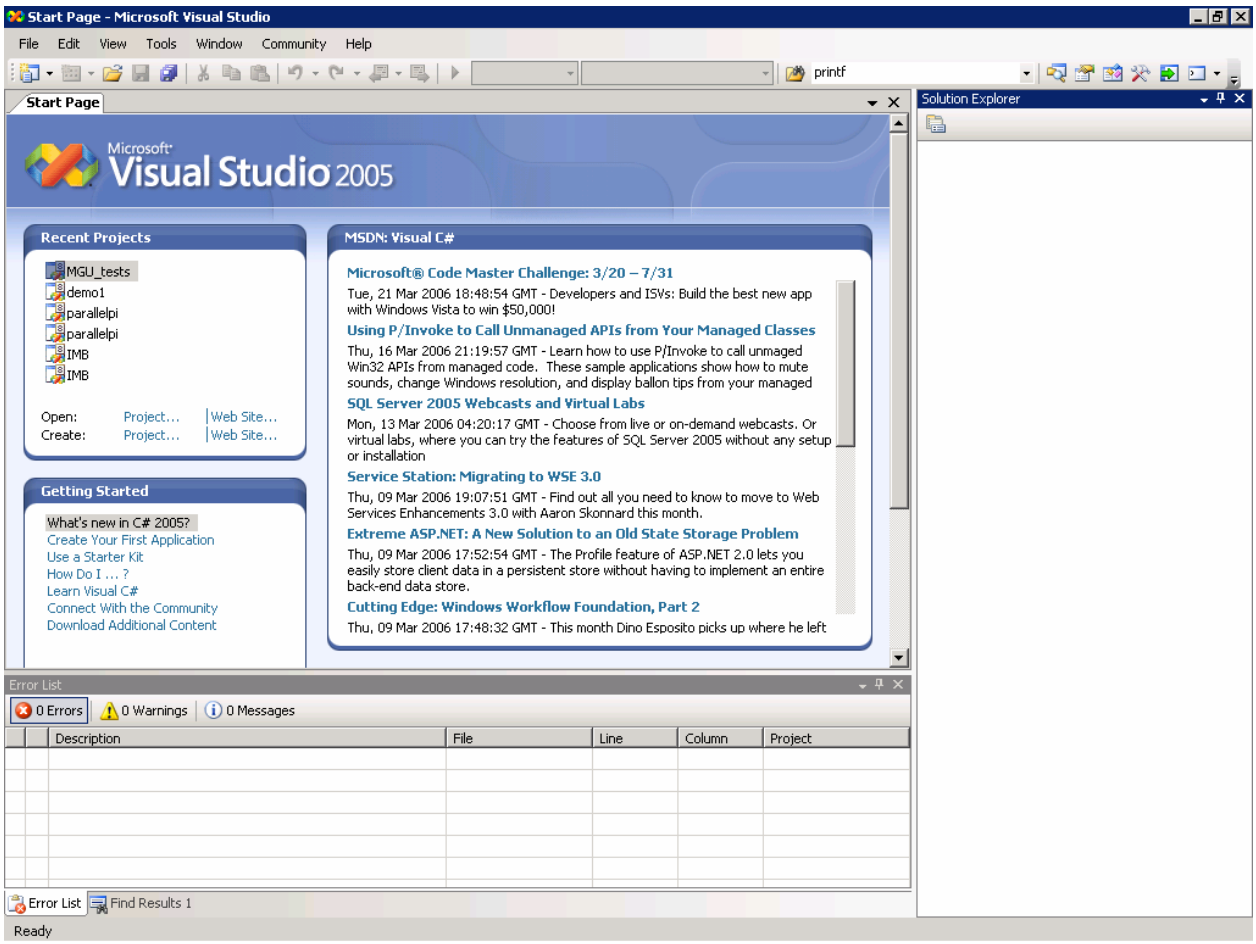
- **The path to the library file msmpi.lib**. Choose the menu option **Project->Project Properties**. In the option **Configuration Properties->C++->Linker->General->Additional Library Directories** enter the path to the library file **msmpi.lib**: **<Installation Directory CCS SDK>\Lib\i386** or **<Installation Directory CCS SDK>\Lib\AMD64** depending on the processor architecture you are using,



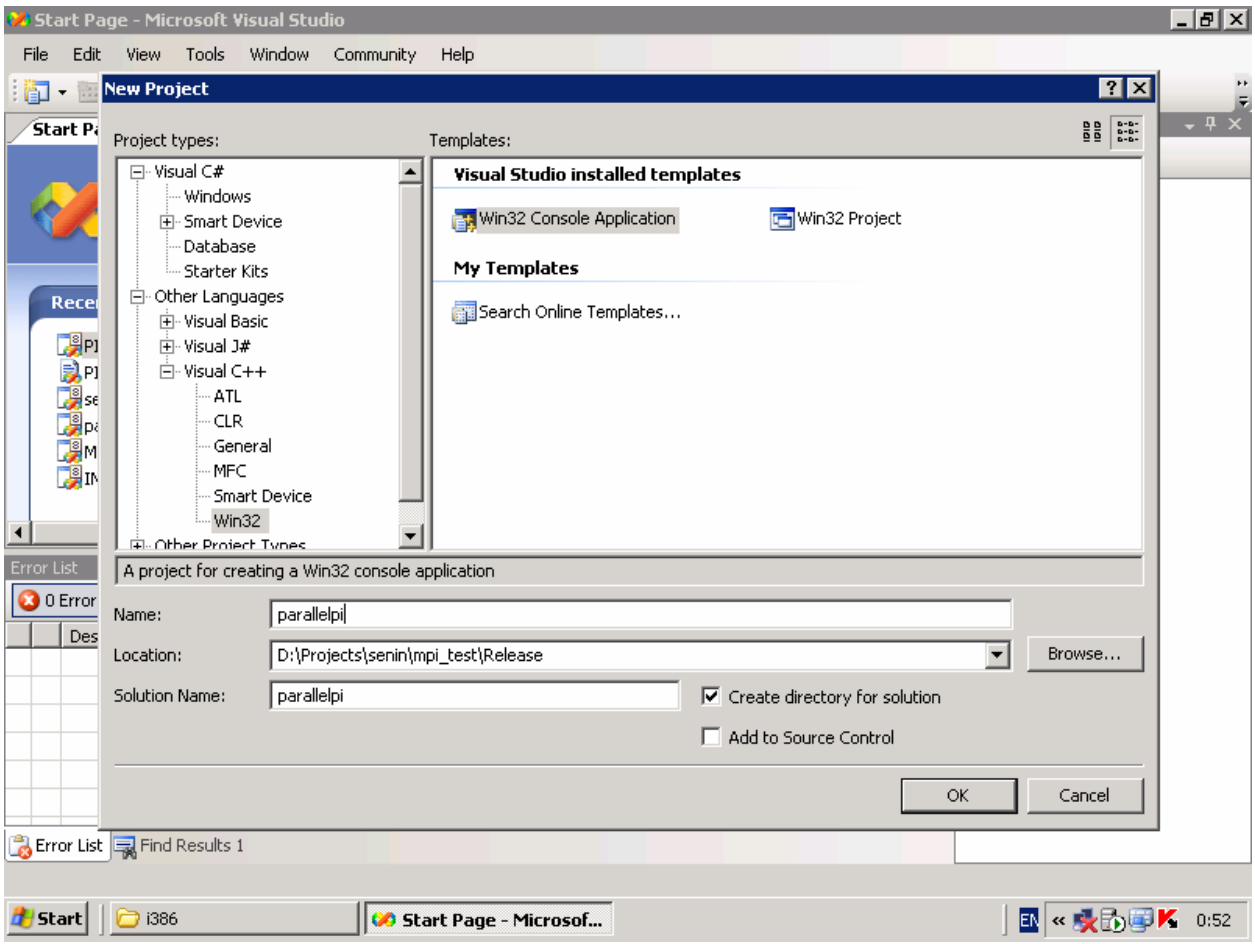
Task 3 – Compiling a Parallel Program in Microsoft Visual Studio 2005

As an example of the parallel program for this task, we will use the parallel algorithm of computing the Pi. In this work we describe only the technical aspects of using Microsoft Compute Cluster Server 2003; the description of the algorithm and the aspects of its implementation are described in Lab “Parallel Programming using MPI”. In this task we will consider only the aspects of using Visual Studio 2005 for compiling a parallel MPI program to be used in the environment MS MPI:

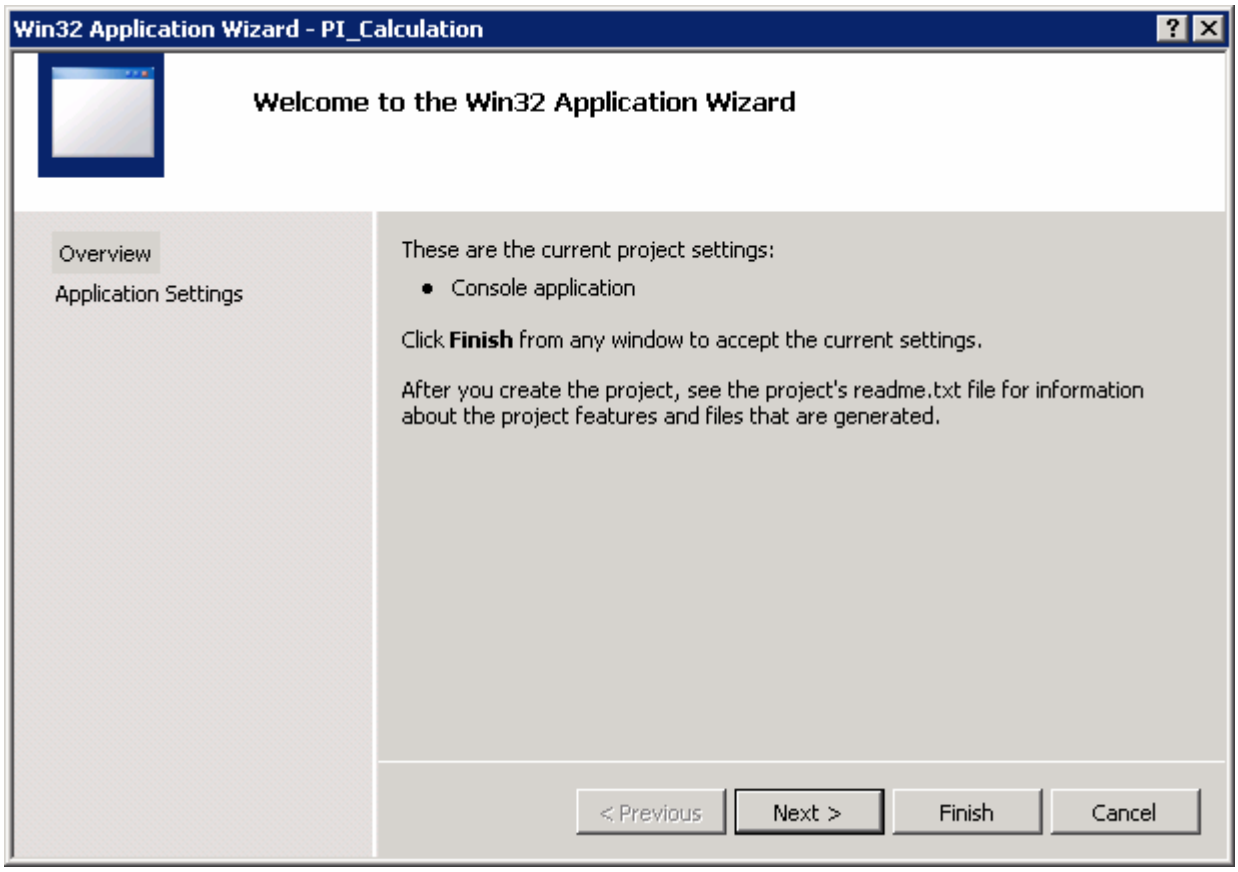
- Run **Microsoft Visual Studio 2005,**



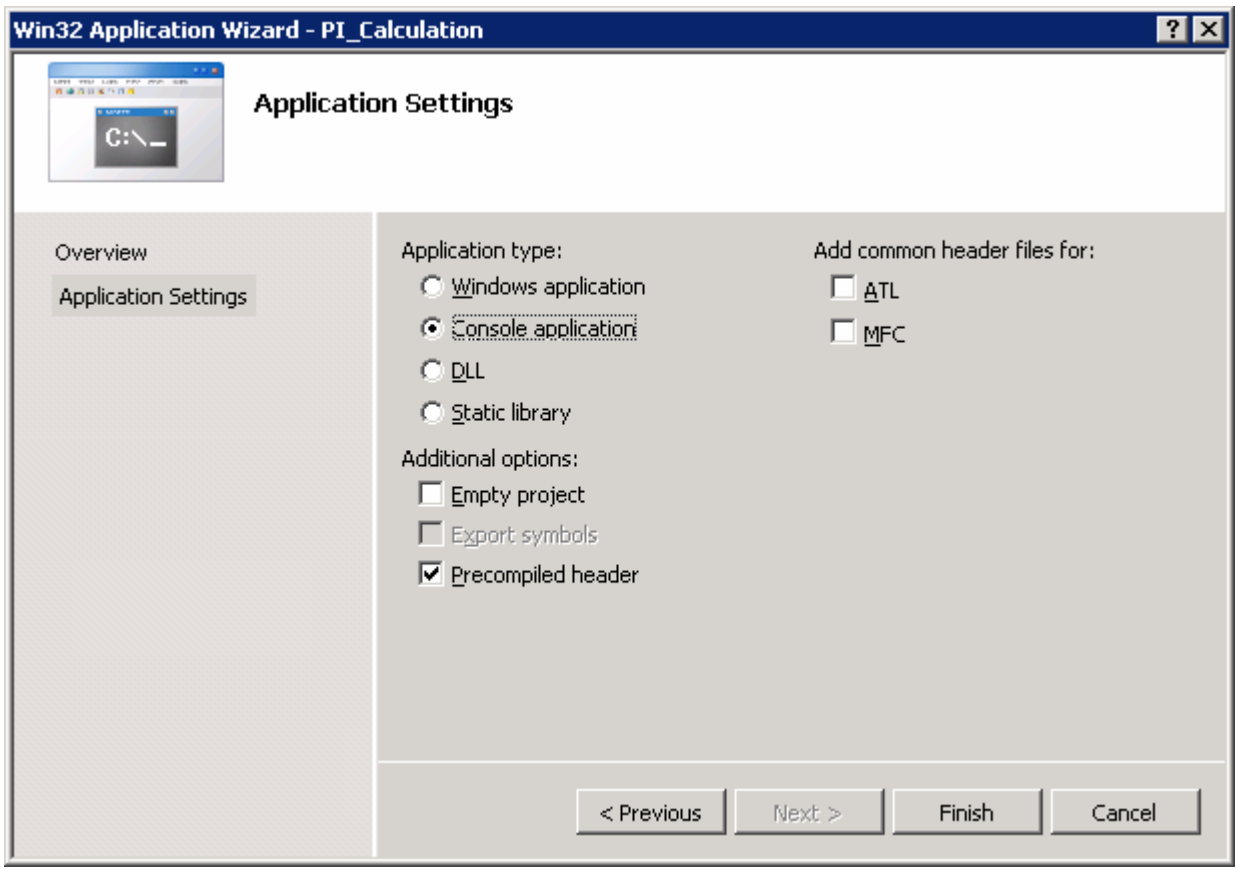
- Create a new project: choose the menu option **File->New->Project**. In the window, where you choose the new project, choose the console **Win32 application** (**Other Languages->Visual C++->Win32->Win32 Console Application**), enter the project name in the field **Name** (for instance, "**parallelpi**") and make sure that the path to the project if chosen correctly (the field **Location**). Press the button **OK** to choose the rest settings of the project being created,



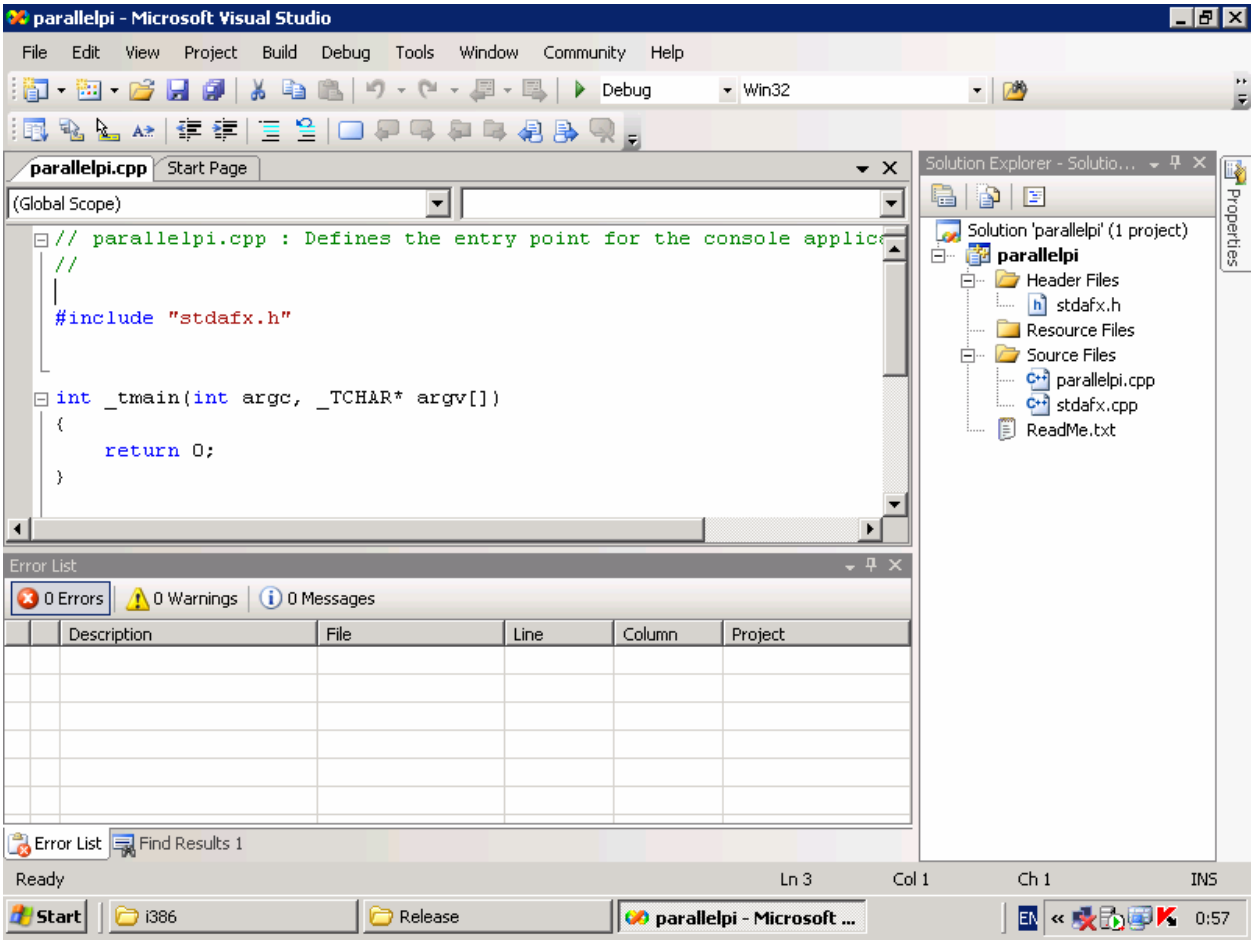
- Press the button **Next** in the new window,



- Choose the project settings in the new window (you can accept all the default settings). Press the button **Finish**,



- In the window **Solution Explorer** double click on the file **parallelpi.cpp** (the file name coincides with the project name you have entered)



- Delete the file content and replace it with the following code (see Lab "Parallel Programming using MPI"):

```
#include "stdafx.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <mpi.h>

void main(int argc, char *argv[]) {
  int     NumIntervals    = 0;   // num intervals in the domain [0,1]
  double  IntervalWidth   = 0.0; // width of intervals
  double  IntervalLength  = 0.0; // length of intervals
  double  IntrvlMidPoint  = 0.0; // x mid point of interval
  int     Interval        = 0;   // loop counter
  int     done            = 0;   // flag
  double  MyPI            = 0.0; // storage for PI approximation results
  double  ReferencePI = 3.141592653589793238462643; // value for comparison
  double  PI;
  char  processor_name[MPI_MAX_PROCESSOR_NAME];
  char  (*all_proc_names)[MPI_MAX_PROCESSOR_NAME];
  int    numprocs;
  int    MyID;
  int    namelen;
  int    proc = 0;

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD,&MyID);
  MPI_Get_processor_name(processor_name,&namelen);

  all_proc_names = (char(*)[128]) malloc(numprocs * MPI_MAX_PROCESSOR_NAME);

  MPI_Gather(processor_name, MPI_MAX_PROCESSOR_NAME, MPI_CHAR,
    all_proc_names, MPI_MAX_PROCESSOR_NAME, MPI_CHAR, 0, MPI_COMM_WORLD);
  if (MyID == 0) {
    for (proc=0; proc < numprocs; ++proc)
      printf("Process %d on %s\n", proc, all_proc_names[proc]);
  }

  IntervalLength = 0.0;
  if (MyID == 0) {
    if (argc > 1) {
      NumIntervals = atoi(argv[1]);
    }
    else {
      NumIntervals = 100000;
    }
    printf("NumIntervals = %i\n", NumIntervals);
  }

  // send number of intervals to all procs
  MPI_Bcast(&NumIntervals, 1, MPI_INT, 0, MPI_COMM_WORLD);

  if (NumIntervals != 0)
  {
    //approximate the value of PI
    IntervalWidth   = 1.0 / (double) NumIntervals;

    for (Interval = MyID+1; Interval <= NumIntervals; Interval += numprocs){
      IntrvlMidPoint = IntervalWidth * ((double)Interval - 0.5);
      IntervalLength += (4.0 / (1.0 + IntrvlMidPoint*IntrvlMidPoint));
    }
    MyPI = IntervalWidth * IntervalLength;
```

```
    // Calculating the sum of all local alues of MyPI
    MPI_Reduce(&MyPI, &PI, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

    //report approximation
    if (MyID == 0) {
      printf("PI is approximately %.16f, Error is %.16f\n",
        PI, fabs(PI - ReferencePI));
    }
  }

  MPI_Finalize();
}
```

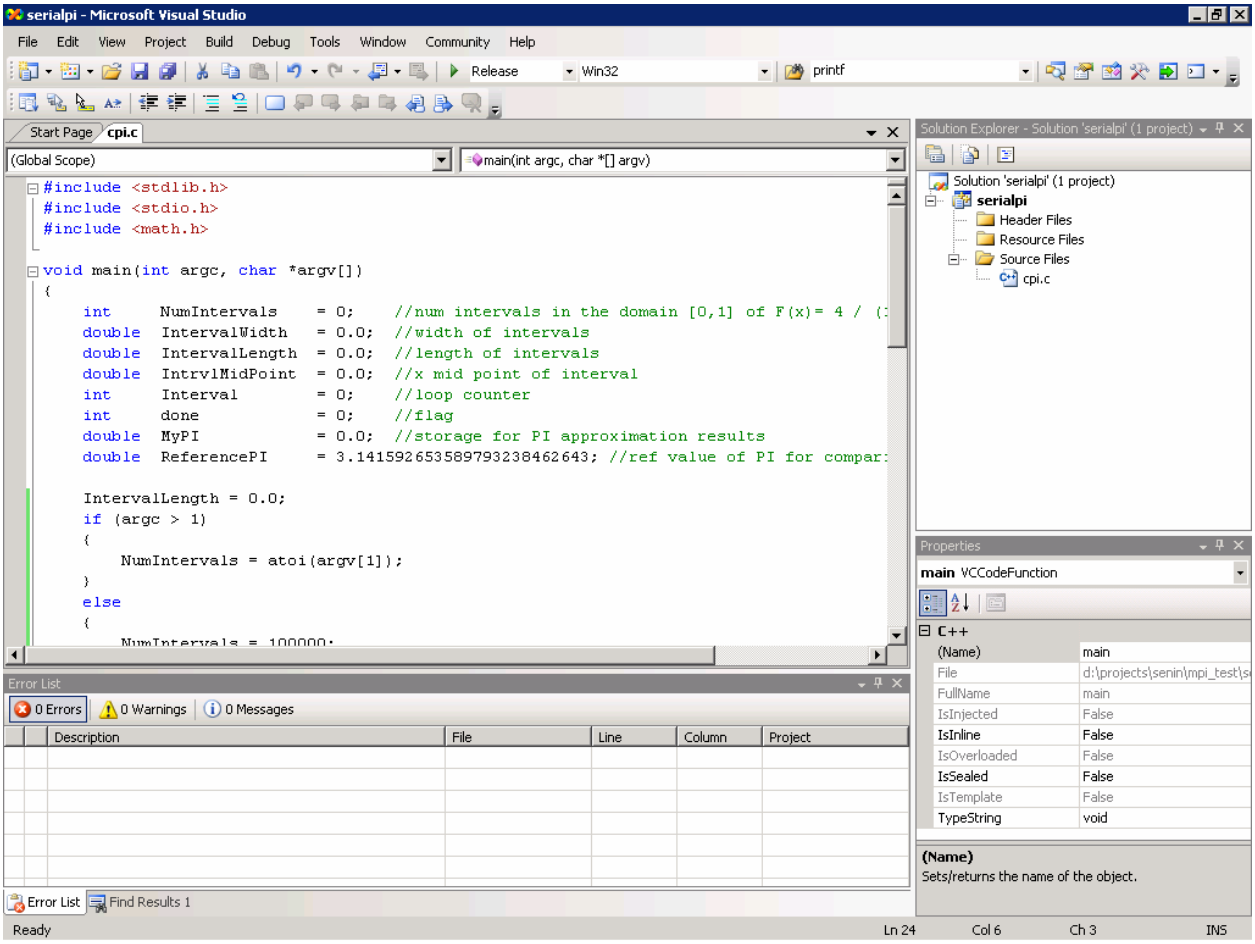
- Make settings of the project Visual Studio 2005 for compiling the MPI part of the project in accordance with the instructions given in "**Setting the development integration environment of Microsoft Visual Studio 2005**",
- Execute the command **Build->Rebuild Solution** for compiling and linking the project,
- Congratulations! The compilation of the program MS MPI is successfully completed.

Exercise 2 – Running a Serial Task

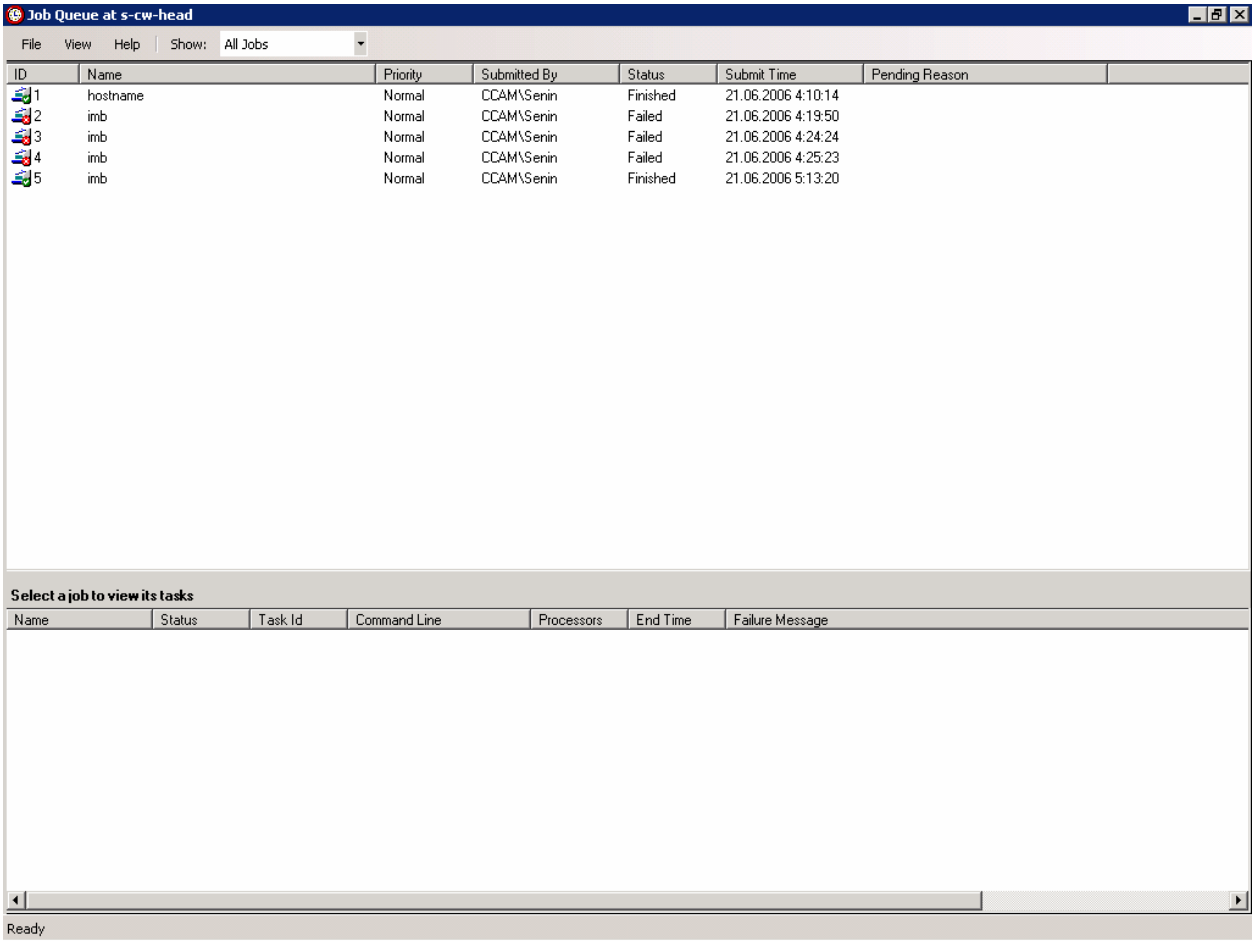
A task is considered to be serial if it uses the resources of only one processor for its execution. Compiling a serial program (and also compiling a parallel program with the use of the technology OpenMP) for using it on the cluster under Compute Cluster Server 2003 does not differ from the standard one and does not require using additional libraries. In the given exercise we will consider all required steps to run a serial task on the cluster.

Launching the Program via Graphic User Interface

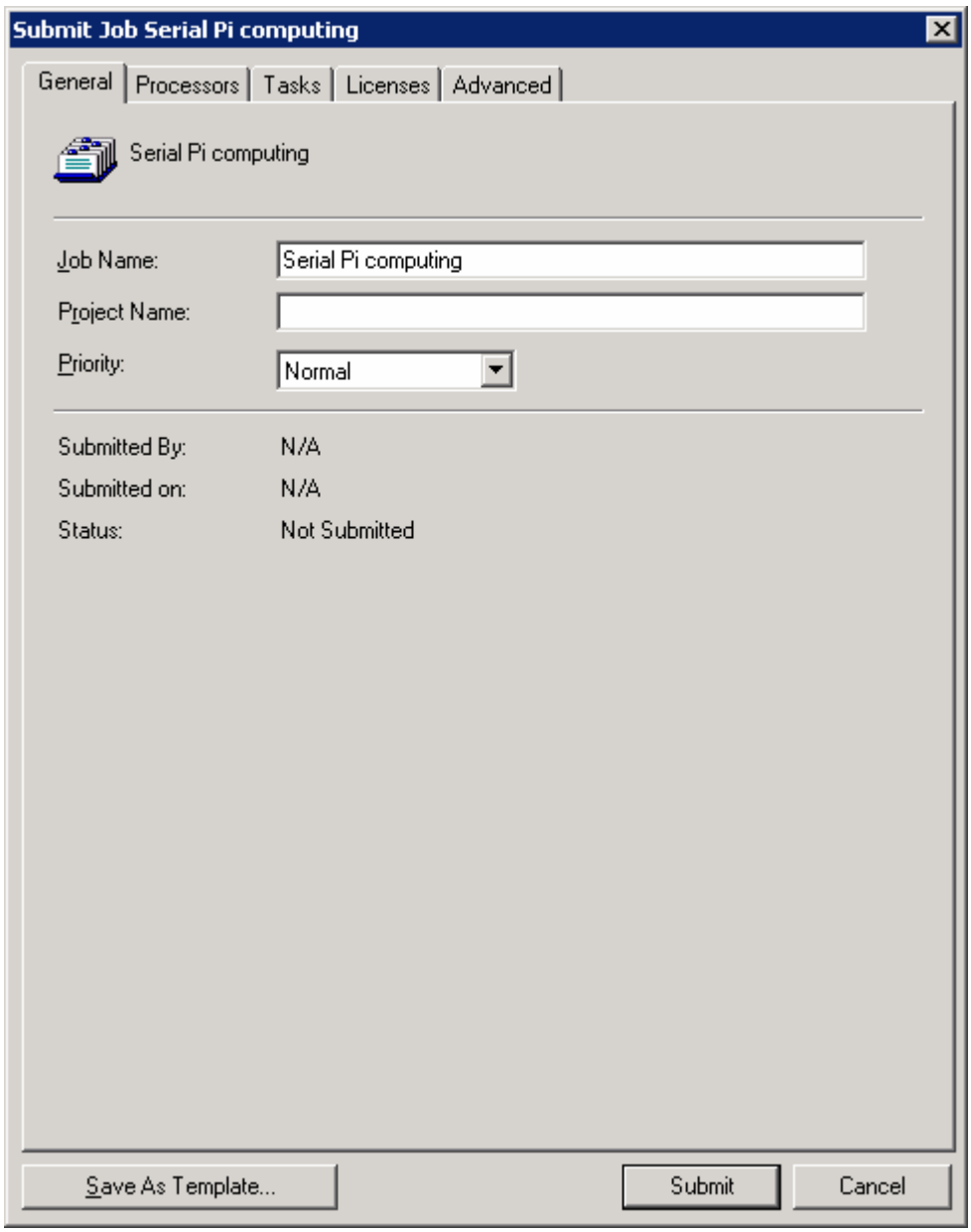
- Open the project of the serial program for calculating the Pi (**serialpi**), which appears together with Lab "Parallel Programming using MPI", and compile the program in the configuration **Release,**



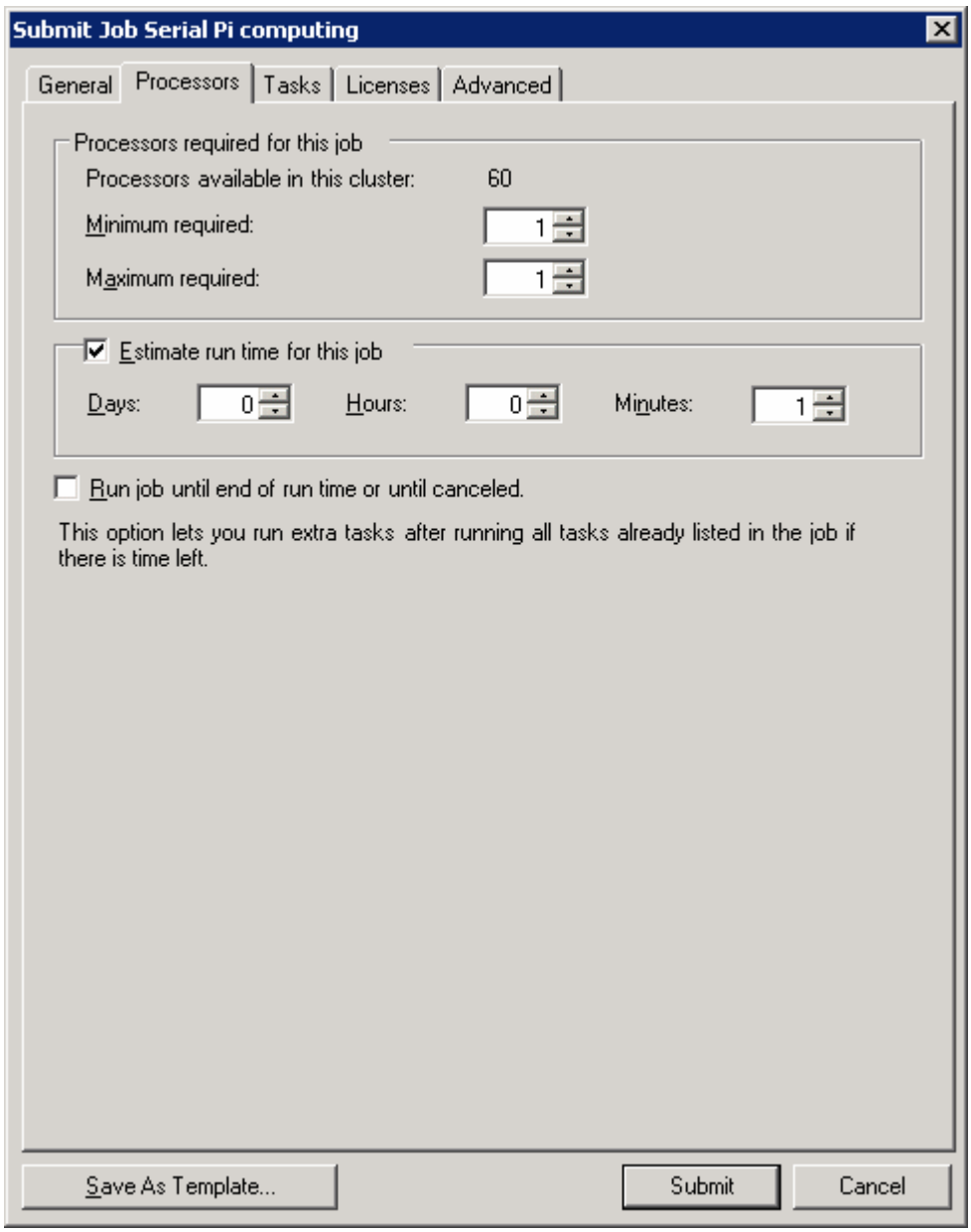
- Open **Computer Cluster Job Manager** (**Start->All Programs->Microsoft Compute Cluster Pack->Compute Cluster Job Manager**) to run the program on the cluster. If you have installed the client part **Compute Cluster Pack** on your PC, you can queue the tasks immediately from your computer, otherwise you should go to the head cluster node or any other node, where the client part is installed, via **Remote Desktop Connection**,



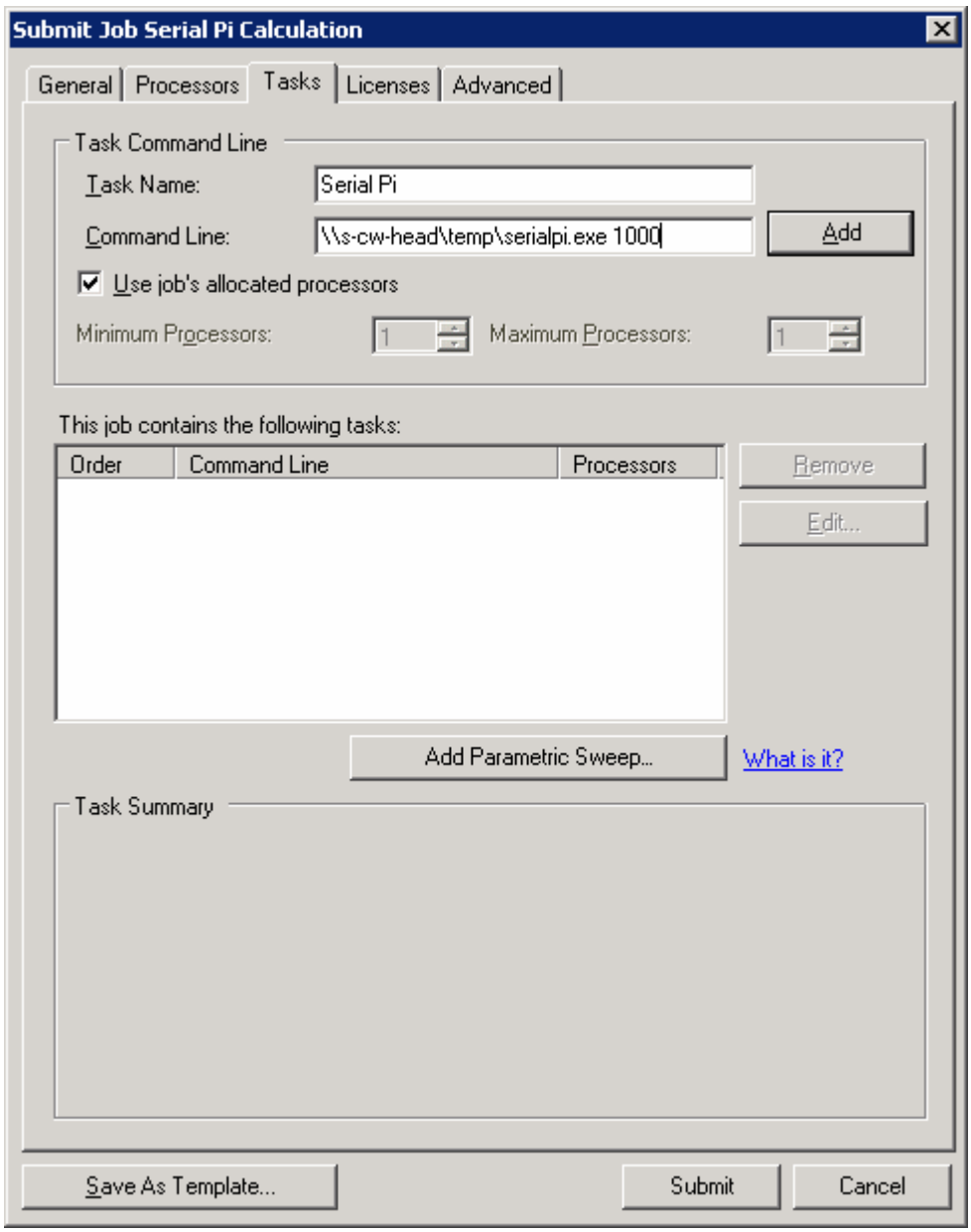
- In the open window of the job manager choose the menu option **File->Submit Job** in order to enqueue the job,
- In the window of job enqueuing enter the name of the job (field **Job Name**), if it is necessary, change the priority of the job (the user's jobs of high priority will be executed prior to the jobs of lower priority). Go to the tab **Processors**,



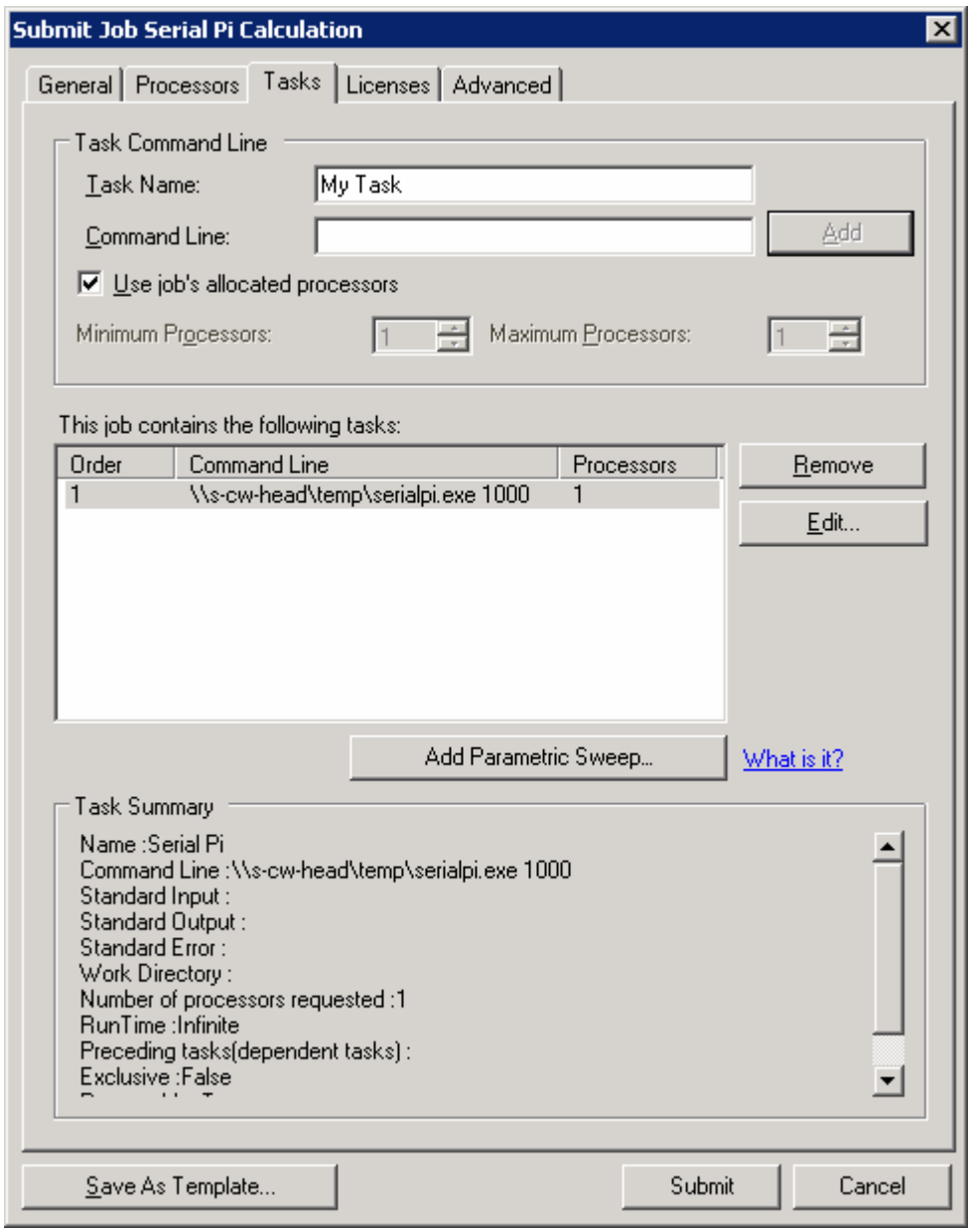
- In the tab **Processors** enter the maximum and the minimum number of the processors required for the jobs (in our case the maximum required number of processors is one, as the task is serial). We assume the number of processors to be maximum if it is optimum for this job (this number of processes will be allocated in case of low cluster load). It is guaranteed that the job will not be started, if the number of available processors on the cluster is less than the minimum number. Additionally you can enter the estimated run time of the job (it will help the job scheduler to distribute the system resources more efficiently) – the window panel **Estimate run time for this job**. If you want the computational resources to be reserved for the job during some specified period of time even after all the tasks have finished their execution, then tick the option **Run job until end of run time or until canceled**. Thus, you will be able to start new tasks of the job even after all the originally given tasks have been completed. Go to the window tab **Tasks** in order to add new tasks,



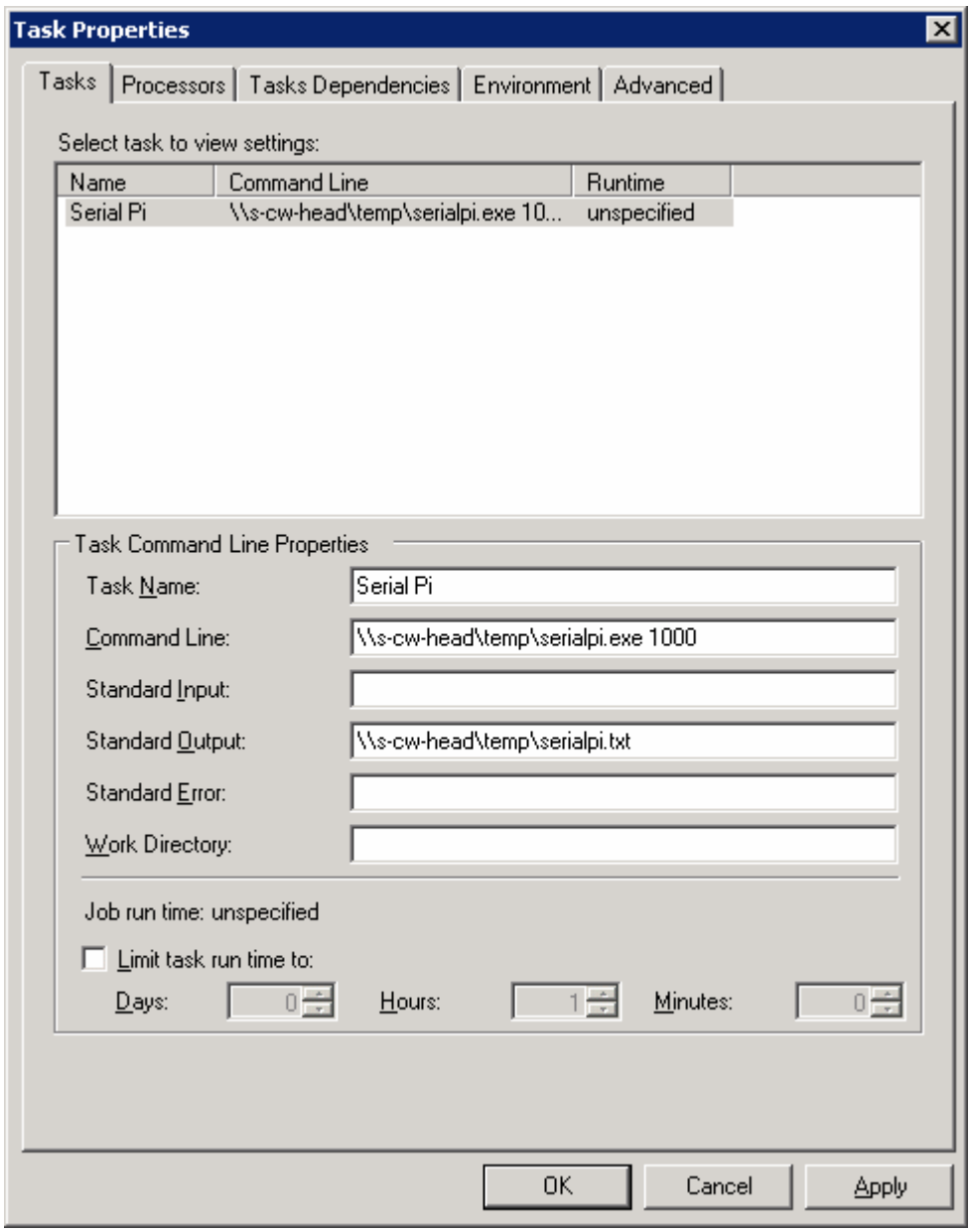
- Enter the task name (the field **Task Name**) and the command for executing (the field **Command Line**) – the program name and the command line parameters. The program should be located on the shared network location, which is accessible from all cluster nodes. Press the button **Add** to add a new task to the job,



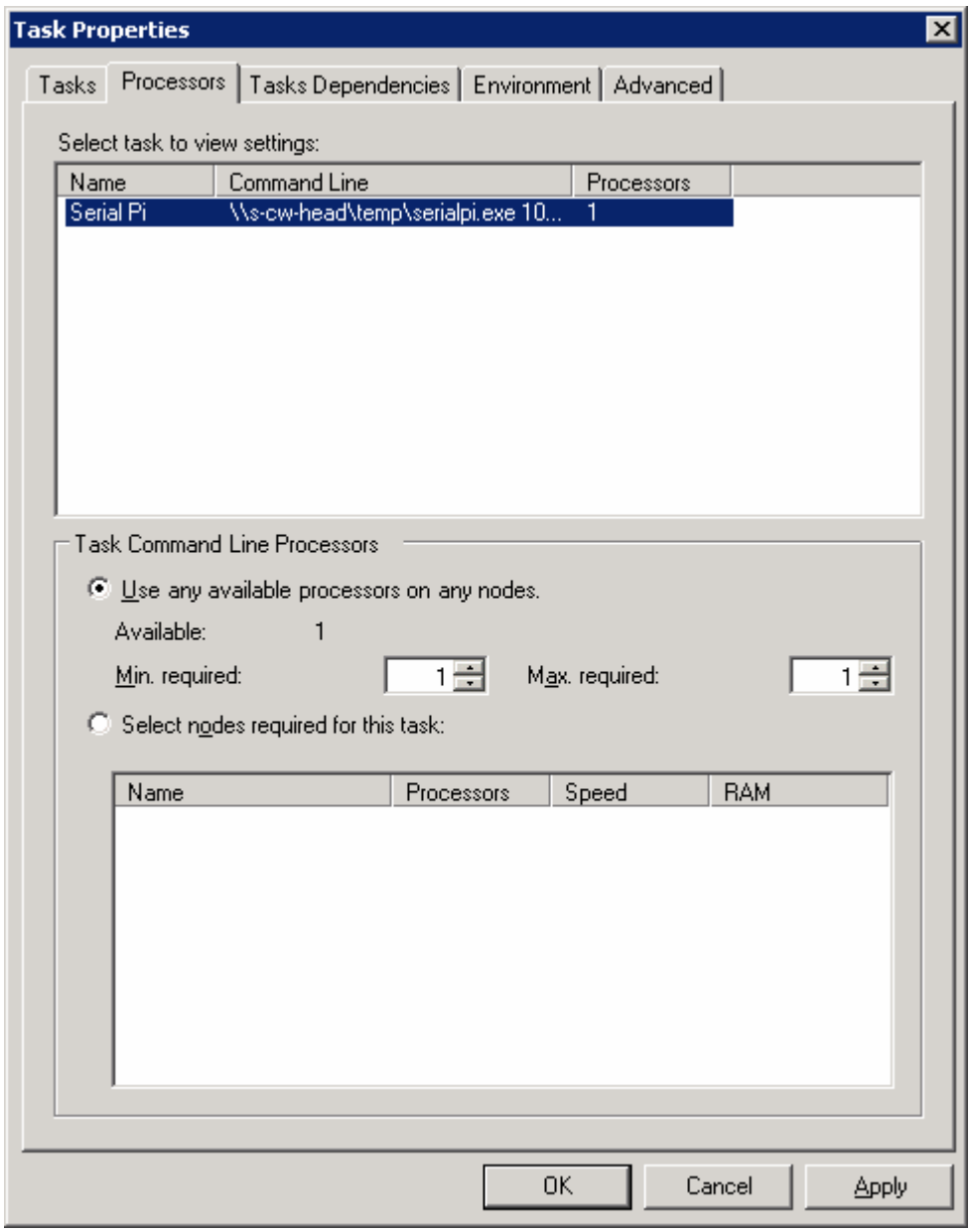
- The task, which was added, will appear in the task list of the current job (the list **This job contains the following tasks**). Select it in the list and press the button **Edit** in order to edit the additional task parameters,



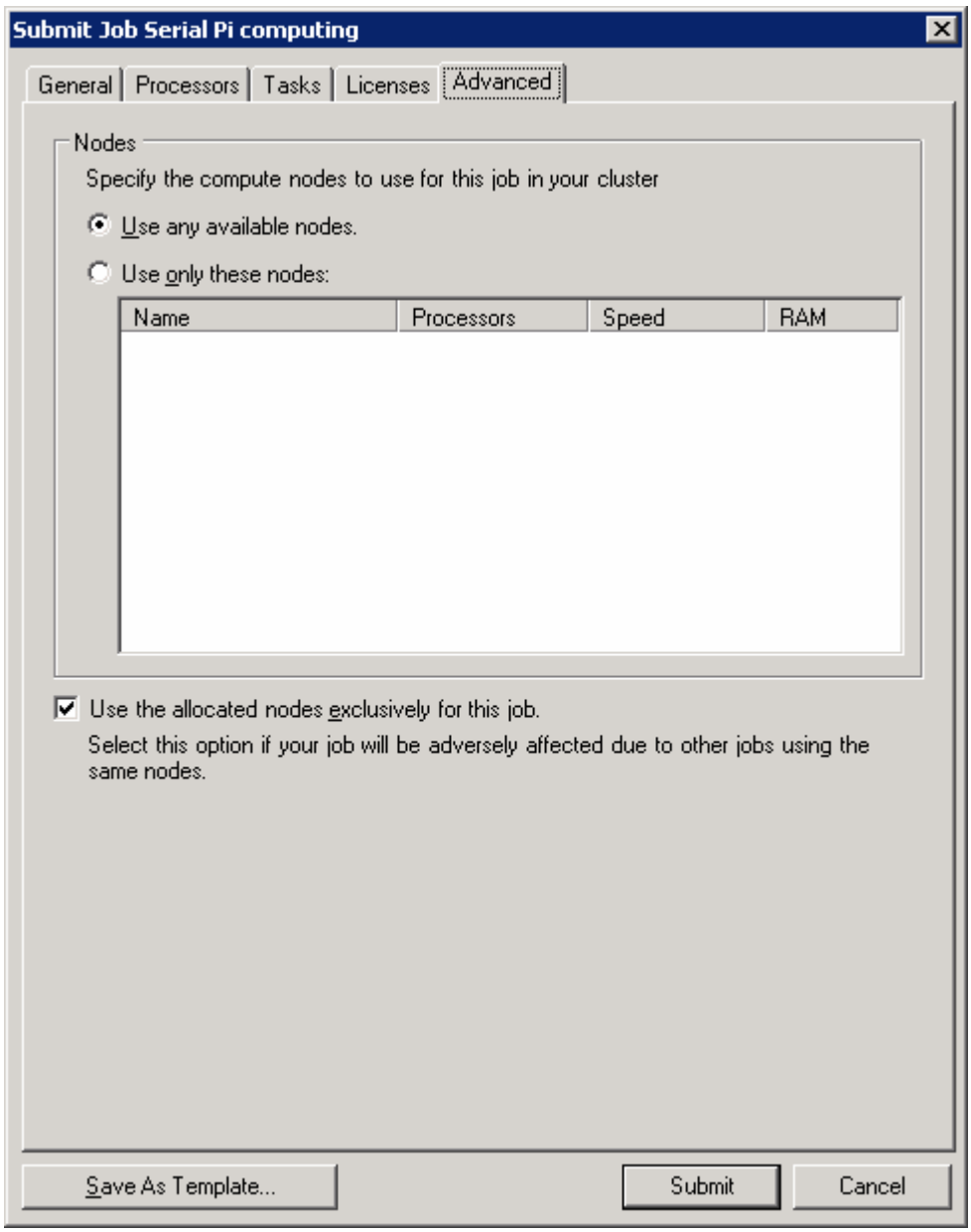
- In the new window enter the file, where you are going to redirect the standard output stream of the console application (the field **Standard Output**). Besides, you may specify the file of the standard input stream (the field **Standard Input**), the file of the standard error stream (the field **Standard Error**), the work directory of the program being executed (the field **Work Directory**) and the time limits on the duration of the task run time (the total run time must not exceed the estimations of the task run time) – the field **Limit task run time to**. Choose the window tab **Processors**,



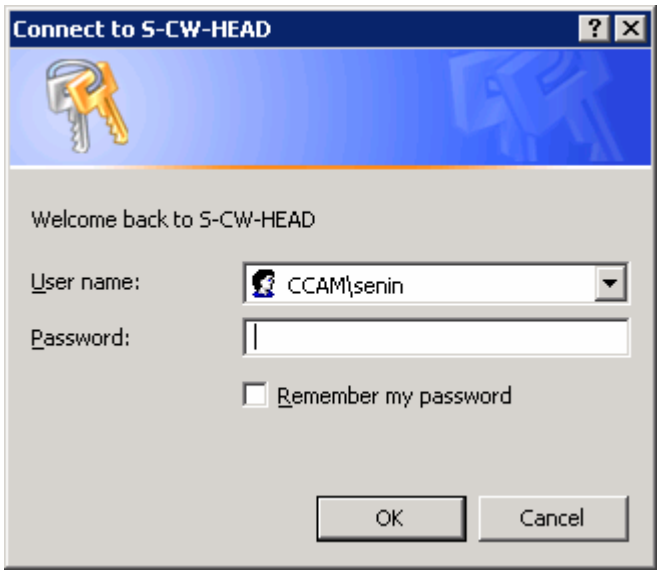
- In the window tab **Processors** in the upper list (**Select task to view settings**) select the task, where you want to change the settings, and specify the minimum and the maximum number of processors for the selected task (the fields **Min. required** and **Max. required**) if you want the job scheduler to select the nodes automatically (**Use any available processors on any nodes**). If you want to select the nodes manually, choose the option **Select nodes required for this task** and tick the required nodes in the lower list. As the task is serial, it requires only one processor. This is the end of setting the task parameters. Press **OK** to save the changes to be done and return to the job setting,



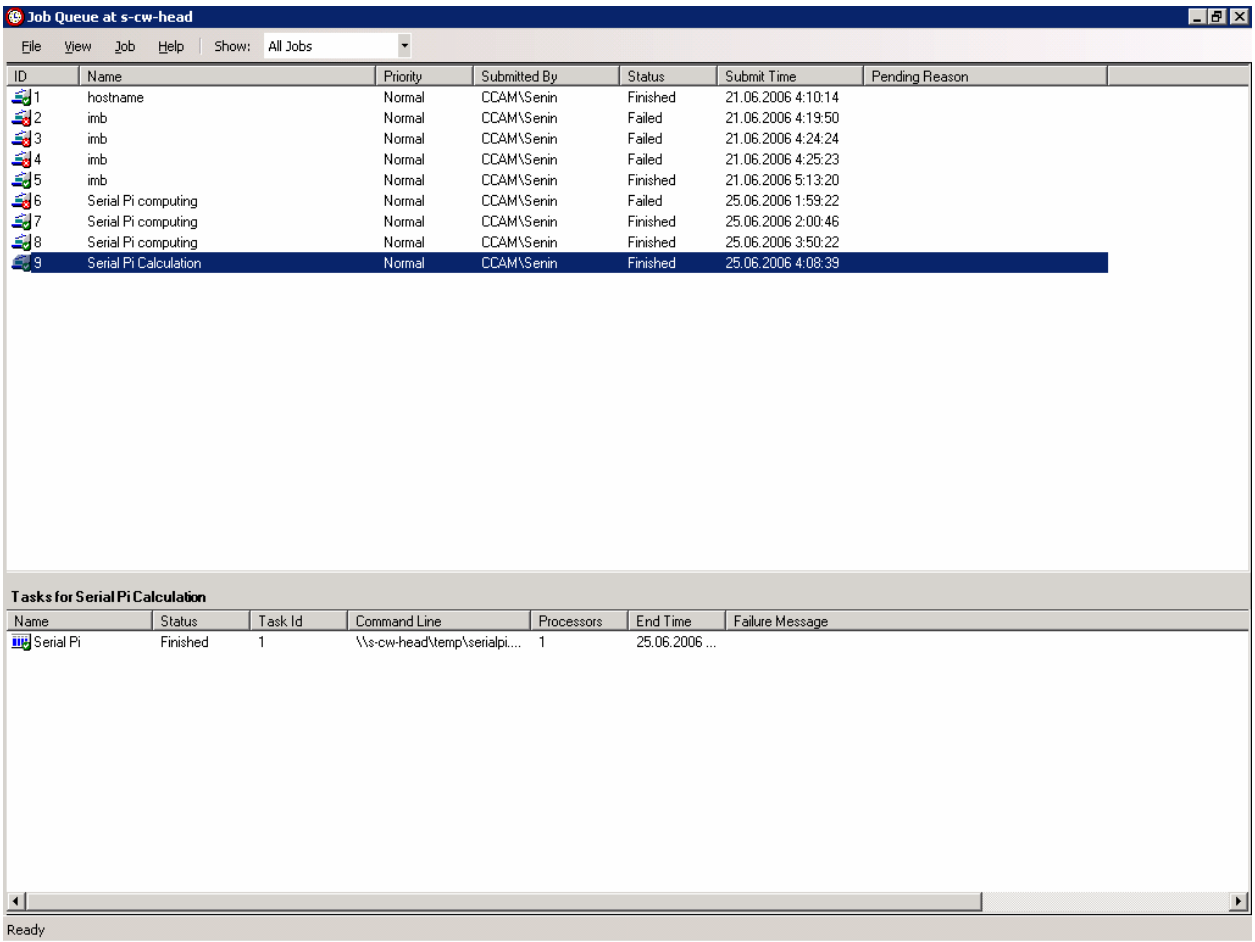
- Go to the window tab **Advanced** and choose the option **Use any available nodes** to choose the nodes for the job automatically. If you want to select the nodes for executing the job manually, choose the option **Use only these nodes**. Remember that if you have selected the nodes manually, these nodes must be also selected for the whole job. Tick **Use the allocated nodes exclusively for this job** in order to prohibit executing several jobs on the same node. Press the button **Submit** to add the job to the queue,



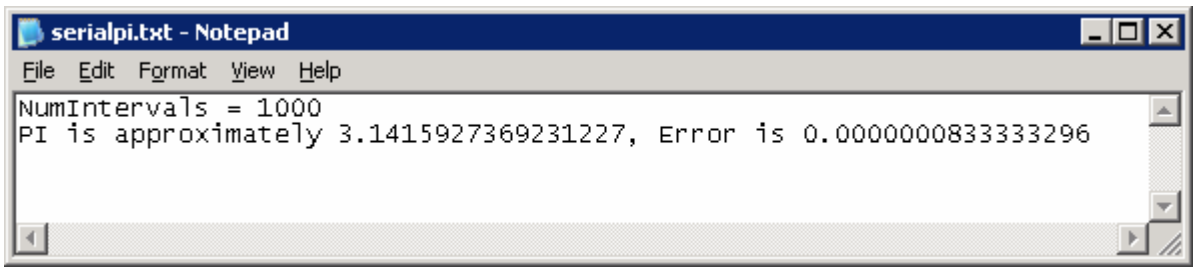
- Enter the name and the password of the user, who is authorized to run tasks on the cluster, and press **OK**,



- The job will appear in the queue. After the execution of the job its status will change to **Finished**.



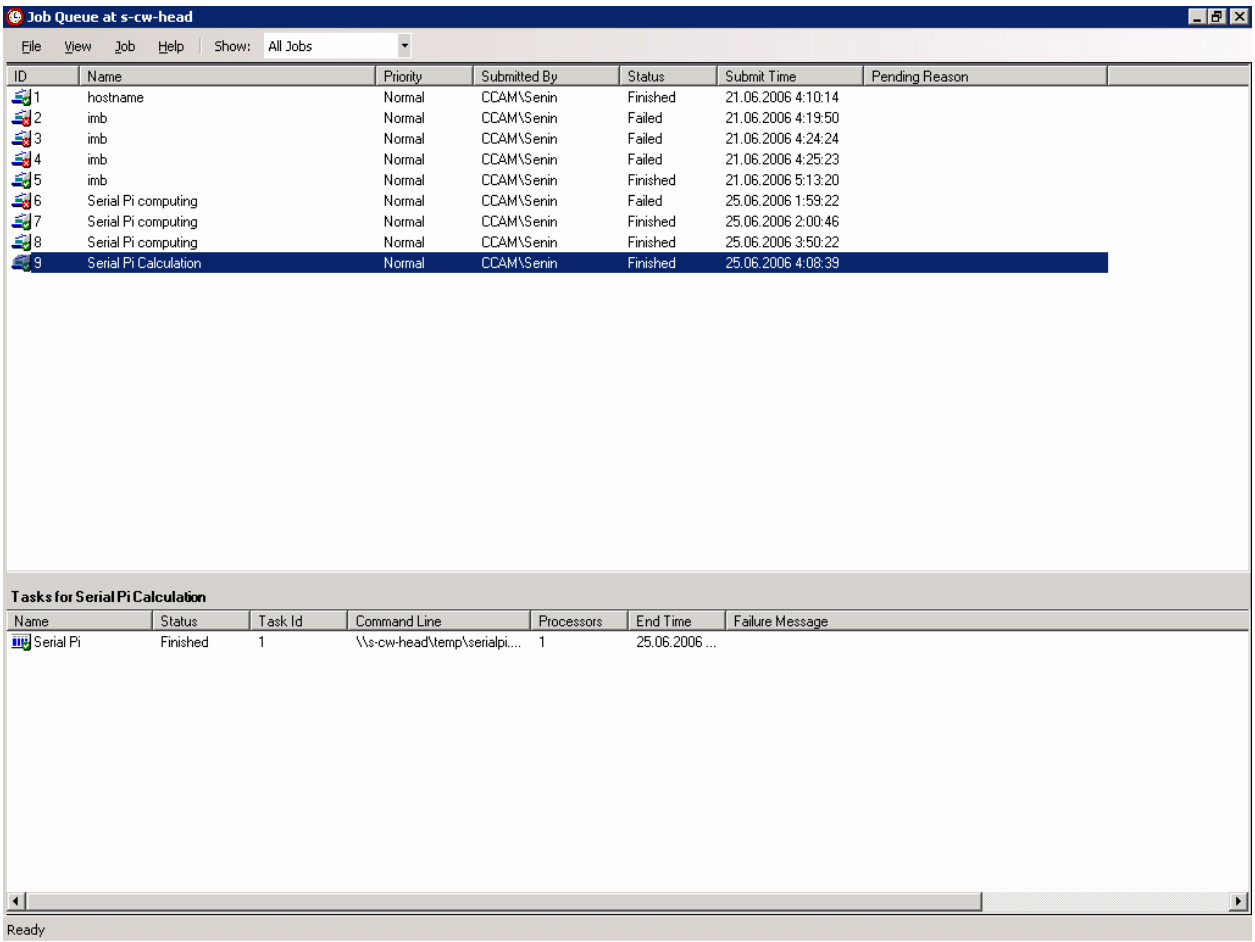
- In the file specified in the job setting for the redirection of the standard output stream, you can find the results of the program execution.



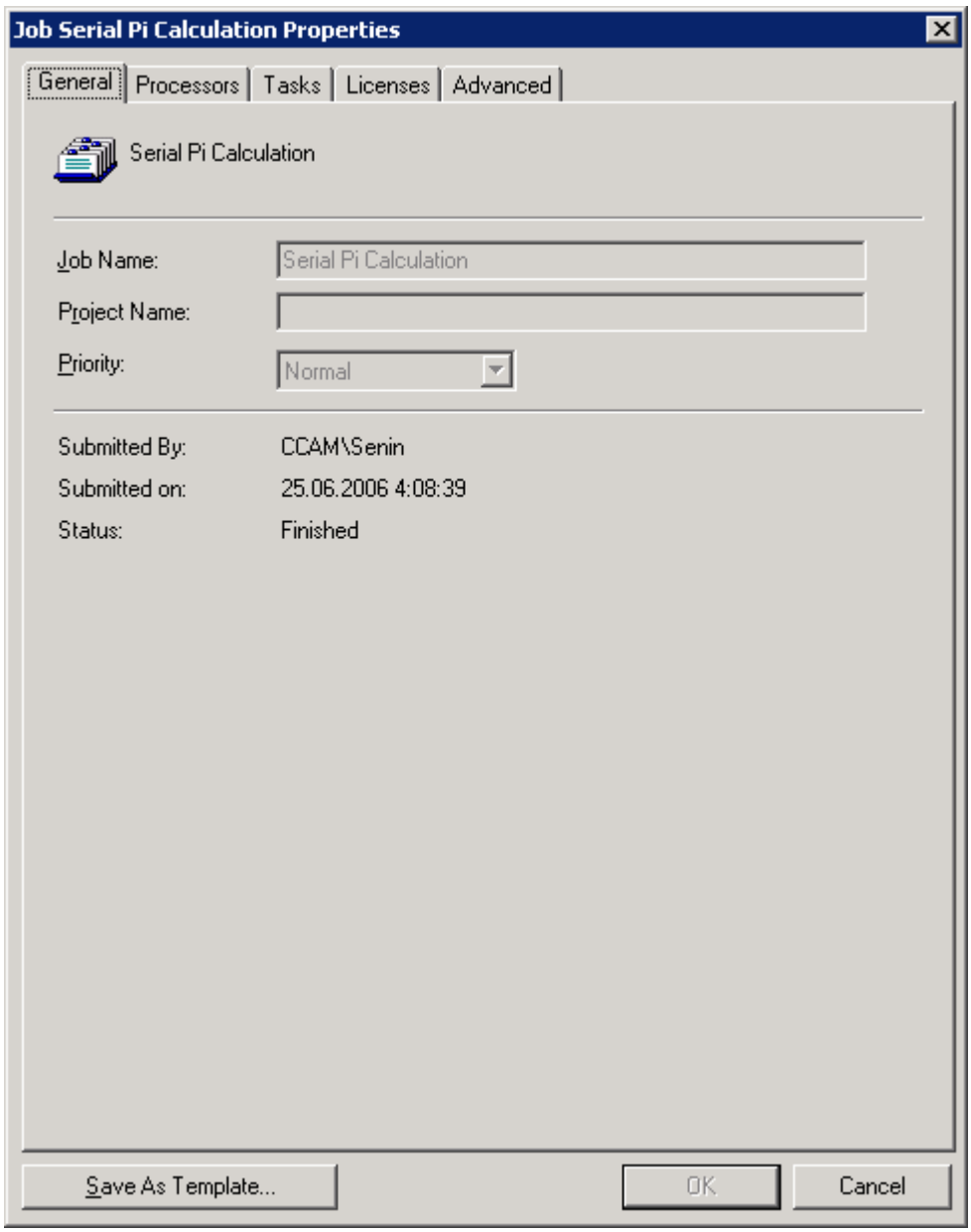
Launching the Program by Means of Template

If you want to launch the task of the serial calculation of the Pi once again (for instance, with other parameters), it will be useful for you to use the command storing all the parameters of the previously launched job in **xml**-file with the possibility to quickly create a copy:

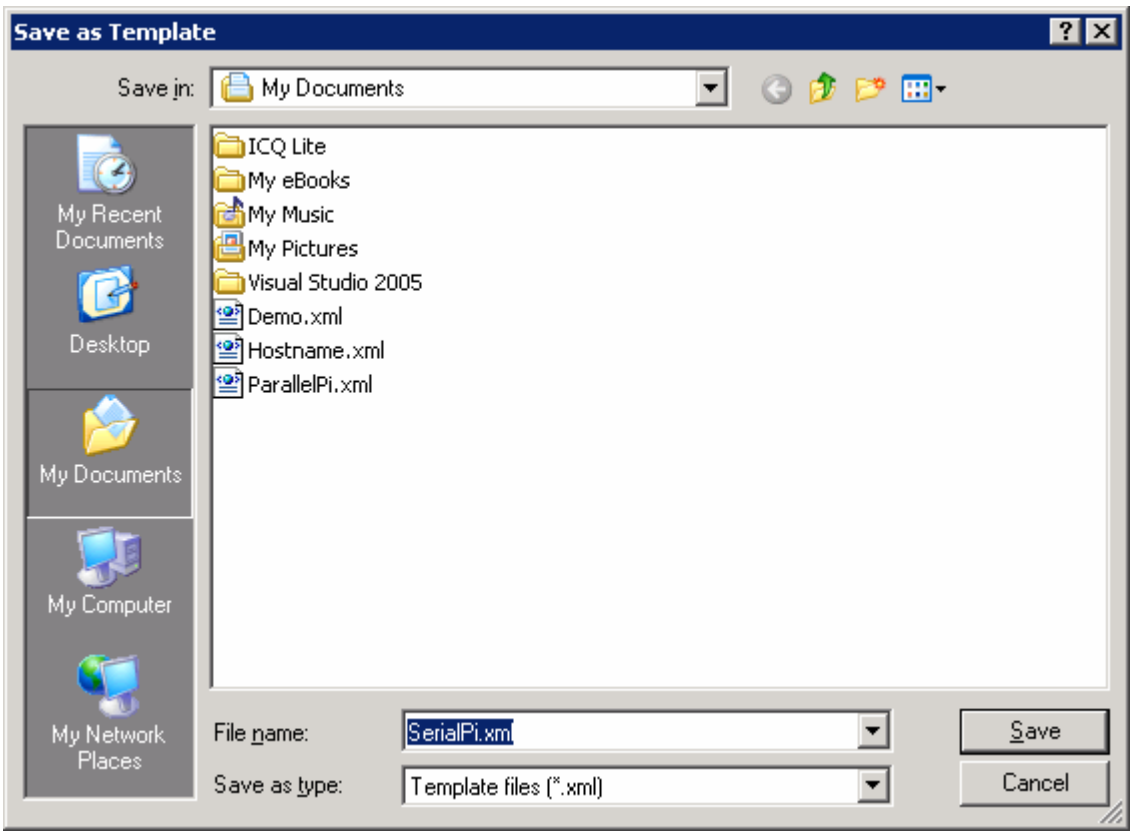
- Open **Computer Cluster Job Manager** (**Start->All Programs->Microsoft Compute Cluster Pack->Compute Cluster Job Manager**) and double click on the job, the parameters of which you want to save in the **xml**-file,



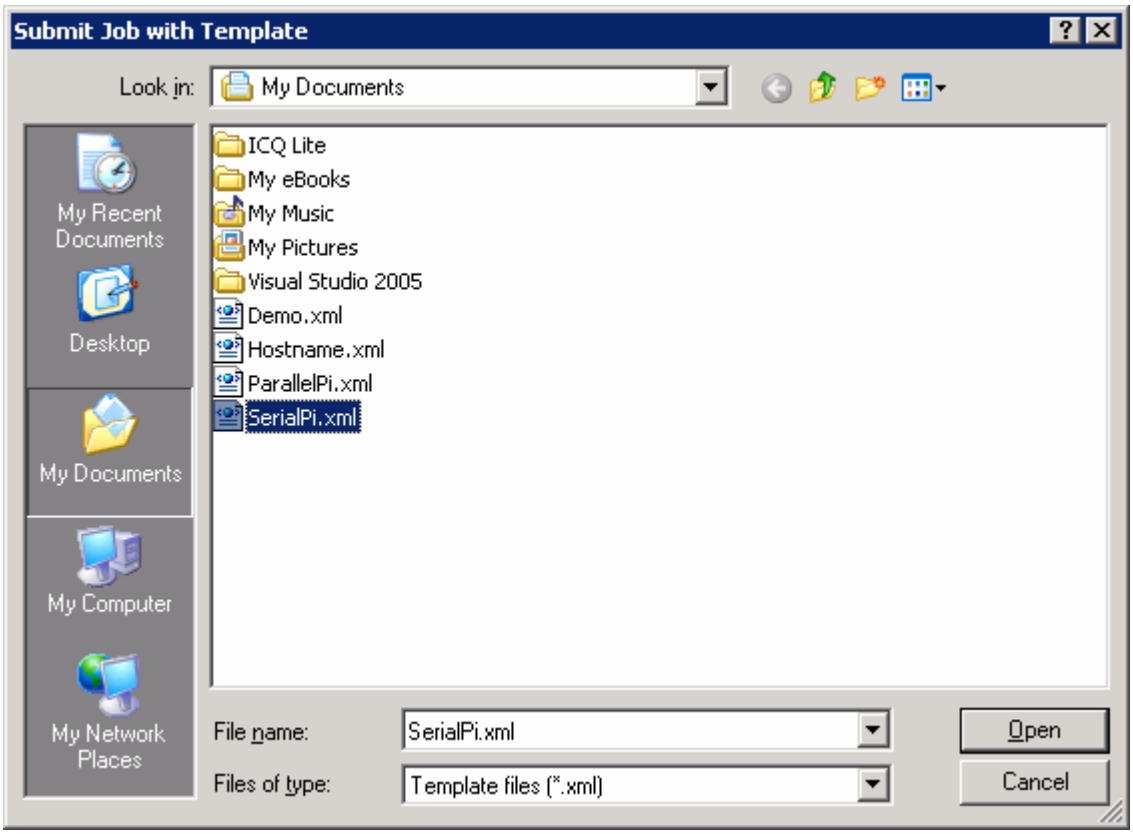
- Press the button **Save As Template** in the window to save the job parameters in the file,



- In the open window choose the directory, where you should save the file, and enter its name. Press the button **Save** to save the job in the file,

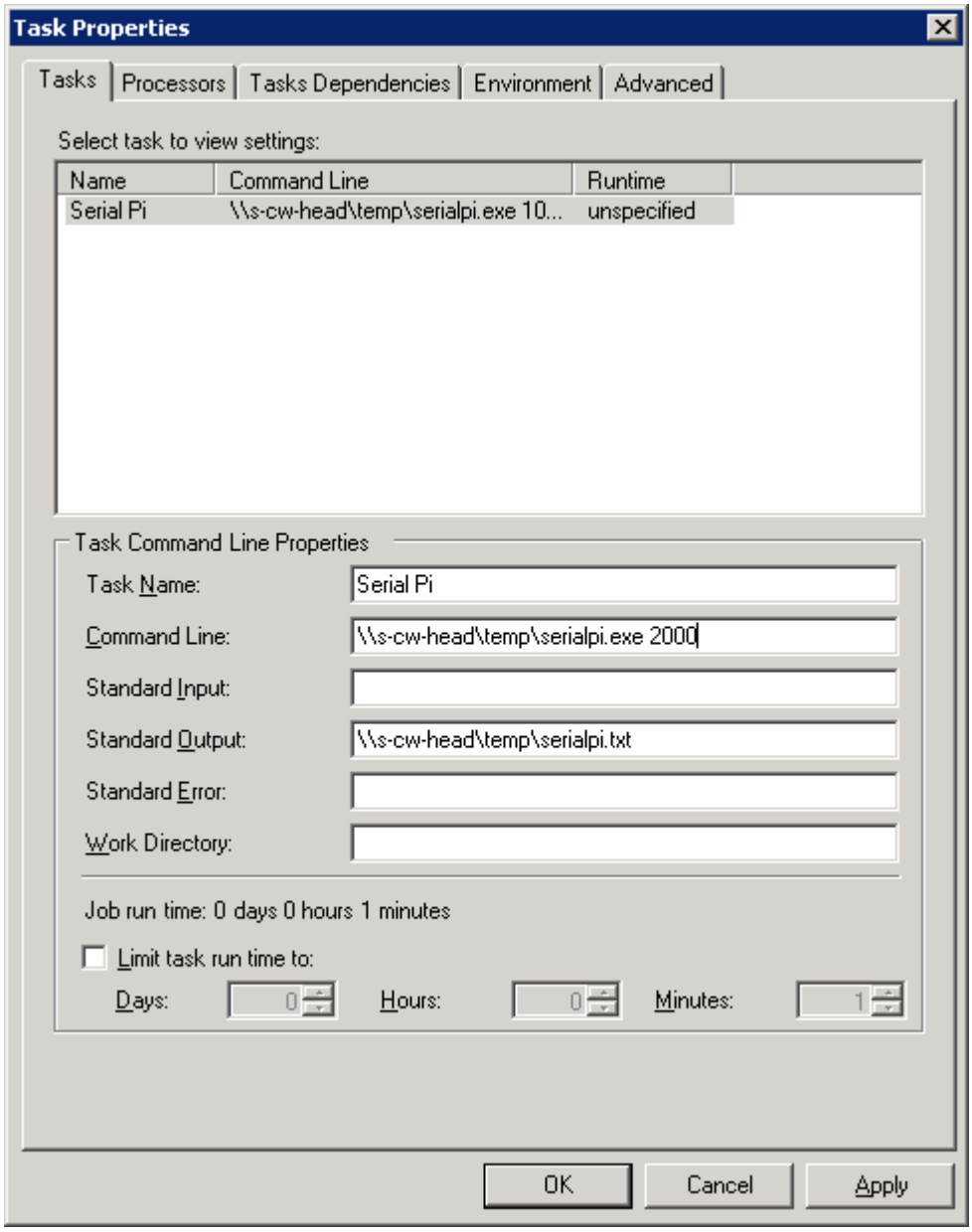


- In order to create the job using the template in the window **Compute Cluster Job Manager,** choose the menu option **File->Submit Job with Template**… In the window for choosing the template, select the file, where you saved the job at the previous step, and press the button **Open**,



- You will see the window for adding the job to the queue. The job parameters and the parameters of all its tasks will be identical to those of the job, which was the basis for creating the template. You can change any required job parameters leaving the rest of them unchanged. Thus, you will save time for editing. For instance, you can increase the number of partitioning the integration interval for calculating the Pi

described in the previous example (the algorithm, which was used for calculating the Pi, is reduced to the numerical calculation of a definite integral), you can leave the rest of the parameters unchanged.



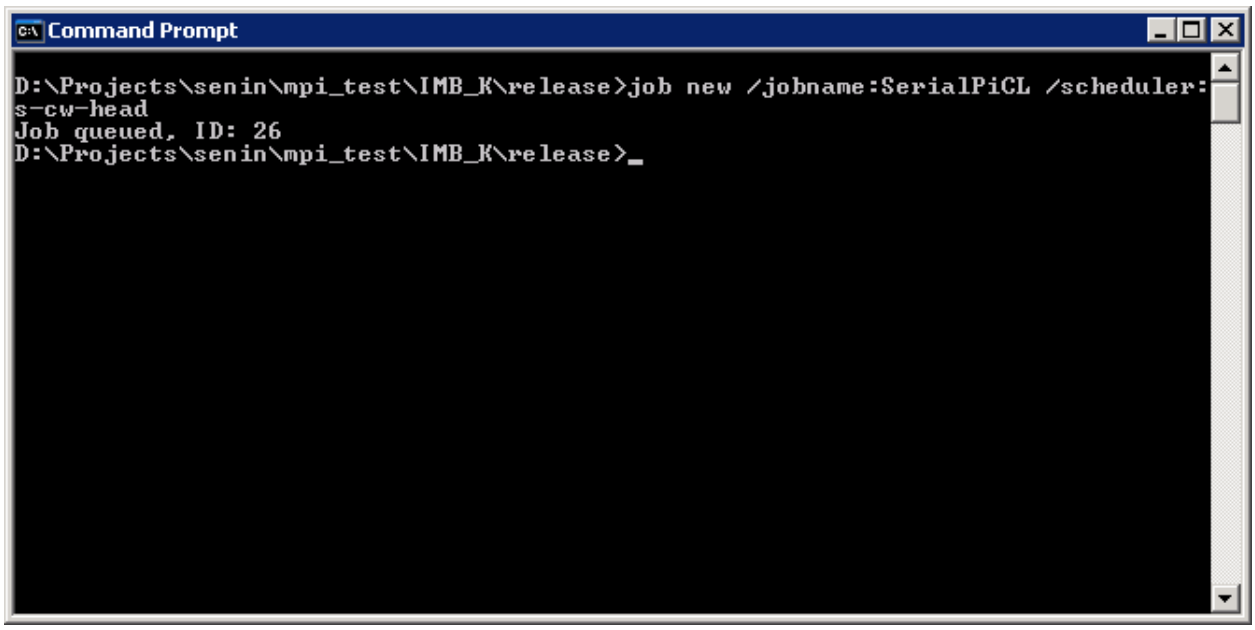
Launching the Program from the Command Line

It is often more convenient to control the course of executing jobs from the command line. Microsoft Compute Cluster Server 2003 includes the utilities, which provide full control over the course of executing jobs on the cluster.

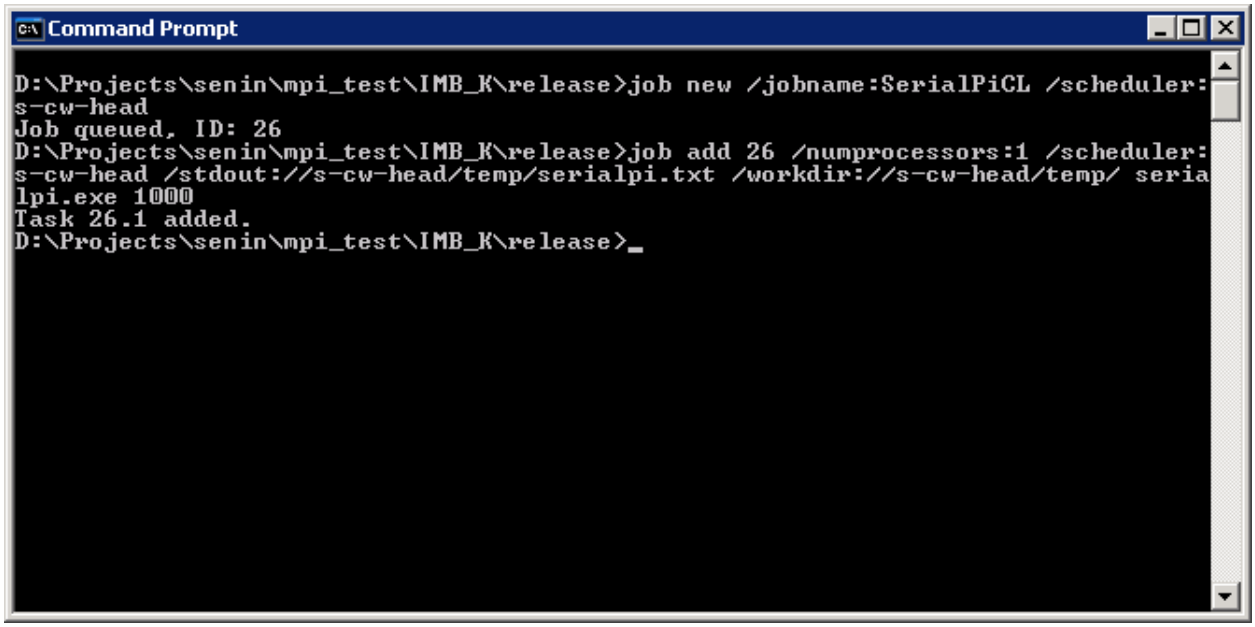
This Lab will illustrate the launch of a serial task from the command line. The launch of a parallel program and the creation of the parametric sweep and a work flow may be also executed from the command line. Additional information on the commands and their parameters is available in the documentation supplied with Microsoft Compute Cluster Pack.

In order to start the serial program of calculating the Pi, you should do the following:

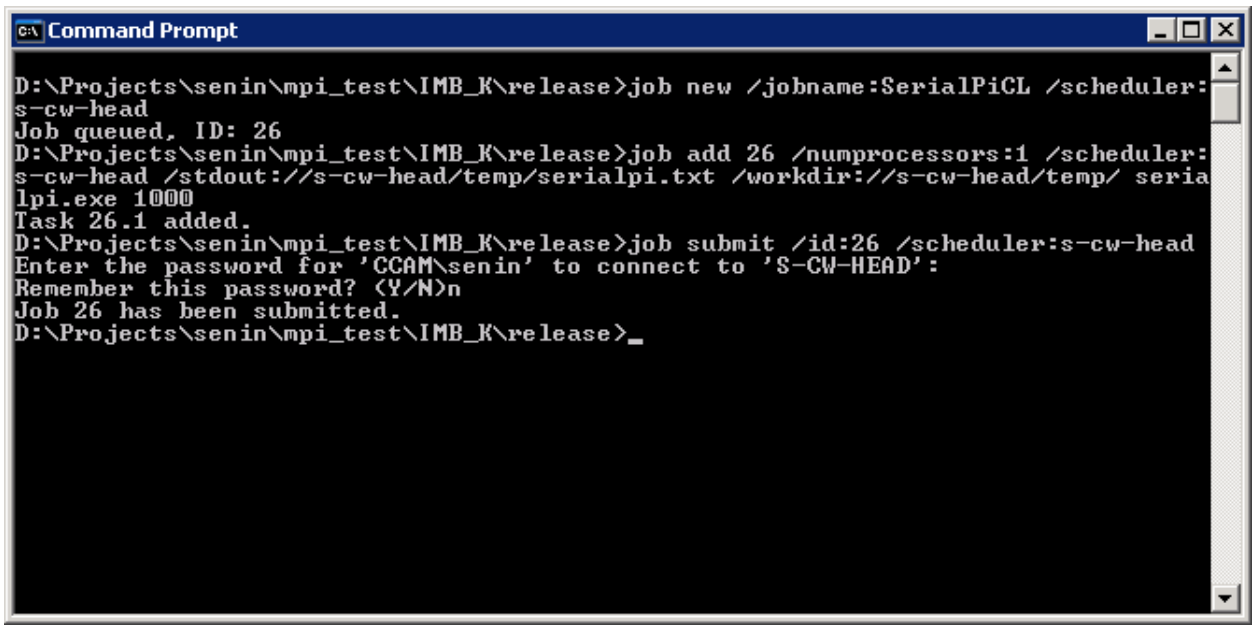
- Open the command window (**Start->Run**, enter the command **cmd** and press **Enter**),
- To create a new job enter the command "**job new /jobname:SerialPiCL /scheduler:s-cw-head**" (do not forget to change the command parameters for the ones, which correspond to your case), where the parameter "**jobname**" is the name of the job being added, "**scheduler**" is the name of the head cluster node. The command will print the identifier (**id**) of the created job. Further on you will work with this **id**,



- In order to add a new task to the job enter the command "**job add 26 /numprocessors:1 /scheduler:s-cw-head /stdout://s-cw-head/temp/serialpi.txt /workdir://s-cw-head/temp/ serialpi.exe 1000**" (do not forget to change the command parameters for the ones, which correspond to your case). Here the number "26" is the job id printed at the previous step. The parameter "**numprocessors**" sets the number of the processors required for the task (to set the minimum and the maximum number of processors you should use the format "**/numprocessors:x-y**", where **x** is the minimum number of processors, and **y** is the maximum number of processors). The parameter "**stdout**" sets the file, where the standard output stream will be redirected. The parameter "**workdir**" sets the directory on default for the application to be launched. After the parameters you will specify the command to launch the application and the arguments of the command line,



- To start scheduling the job enter the command "**job submit /id:26 /scheduler:s-cw-head**" (do not forget to change the command parameters for the ones, which correspond to your case). Enter the user's password, which you used to login in the system. If requested whether to store your password in order not to enter it further, enter "**n**" to refuse,

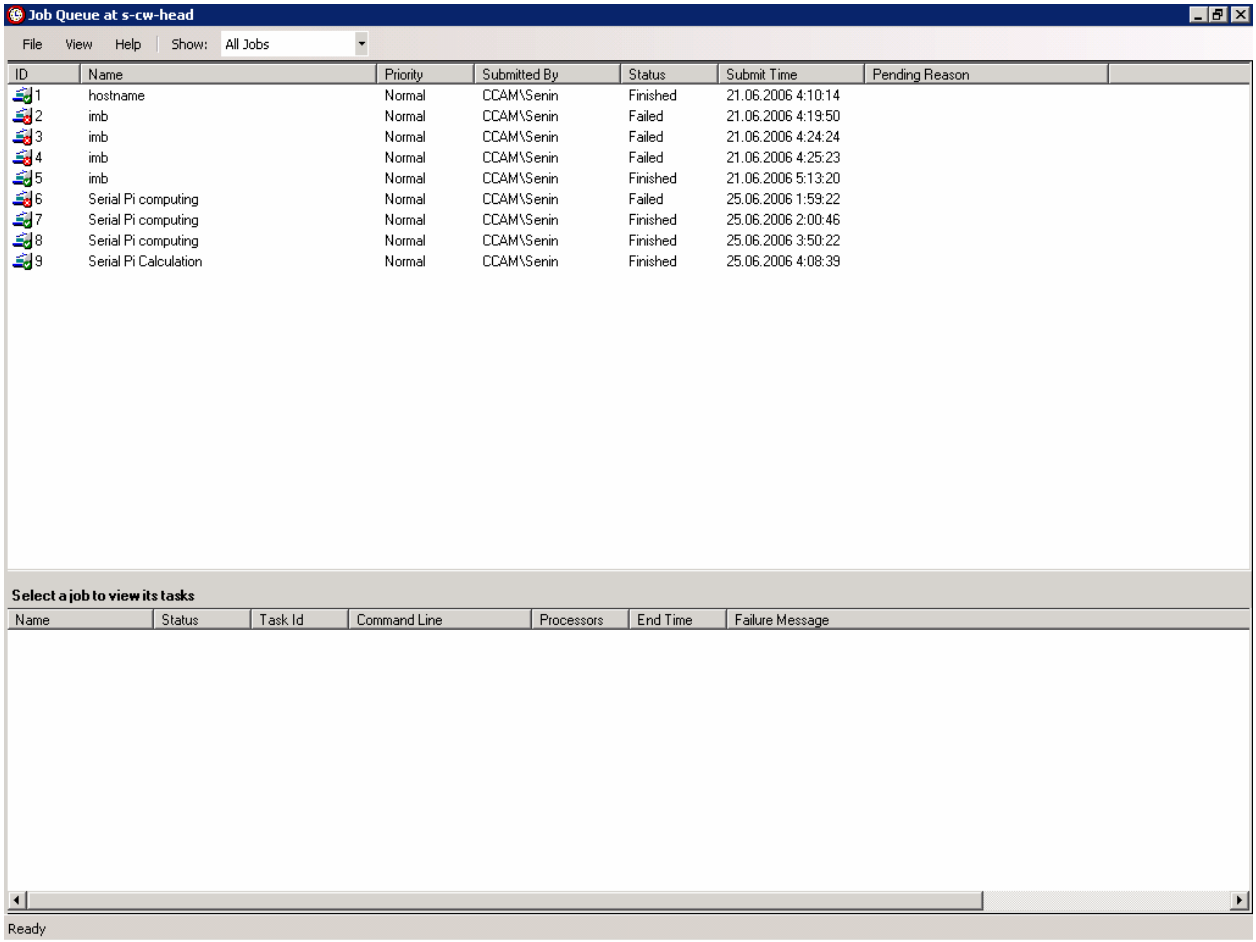


- Your task has been added to the queue, and the scheduler has begun to plan its launch. You can track the status in the program **Job Manager** or by entering the command "**job list /scheduler:s-cw-head /all**" (do not forget to change the command parameters for the ones, which correspond to your case).

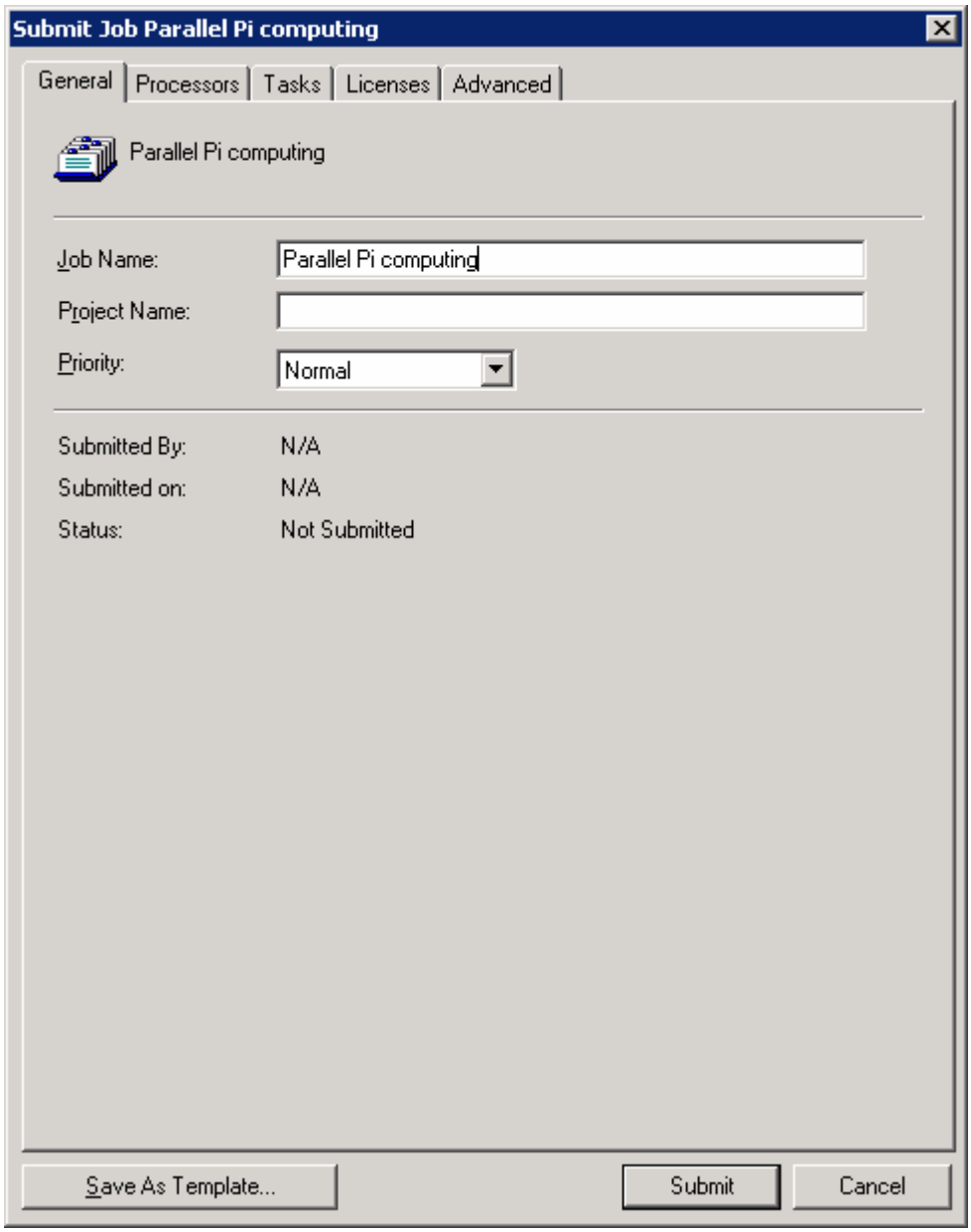
Exercise 3 – Launching a Parallel Job

In Task 3 of Exercise 1 we compiled the parallel program for calculating the value of Pi for MS MPI. Let us launch it on the cluster under Microsoft Compute Cluster Server 2003:

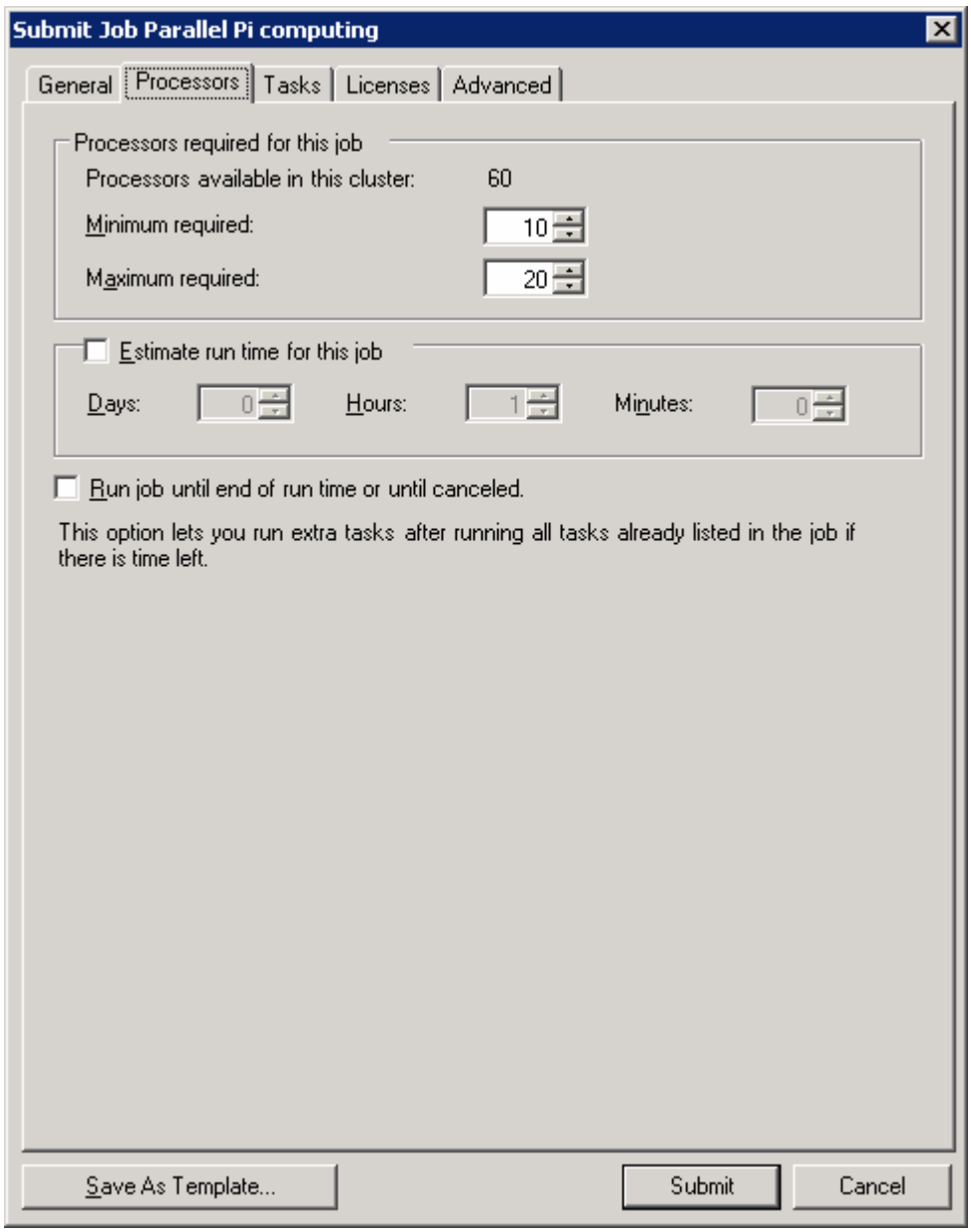
- Open **Computer Cluster Job Manager** (**Start->All Programs->Microsoft Compute Cluster Pack->Compute Cluster Job Manager**) to start the program on the cluster,



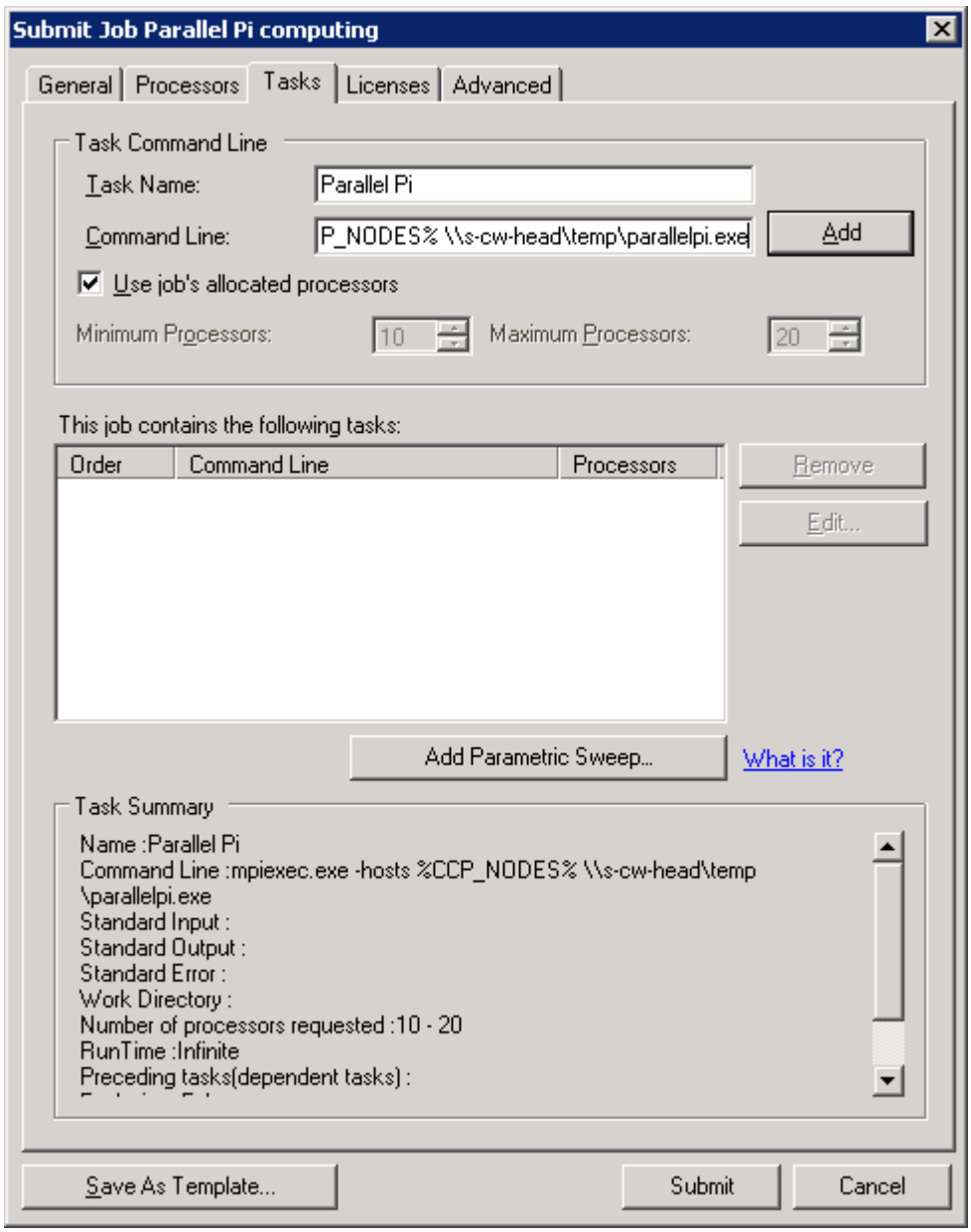
- In the window of the job manager choose the menu option **File->Submit Job** in order to add a new job to the queue,
- In the window of adding the job to the queue enter the job name (the field **Job Name**). Go to the window tab **Processors**,



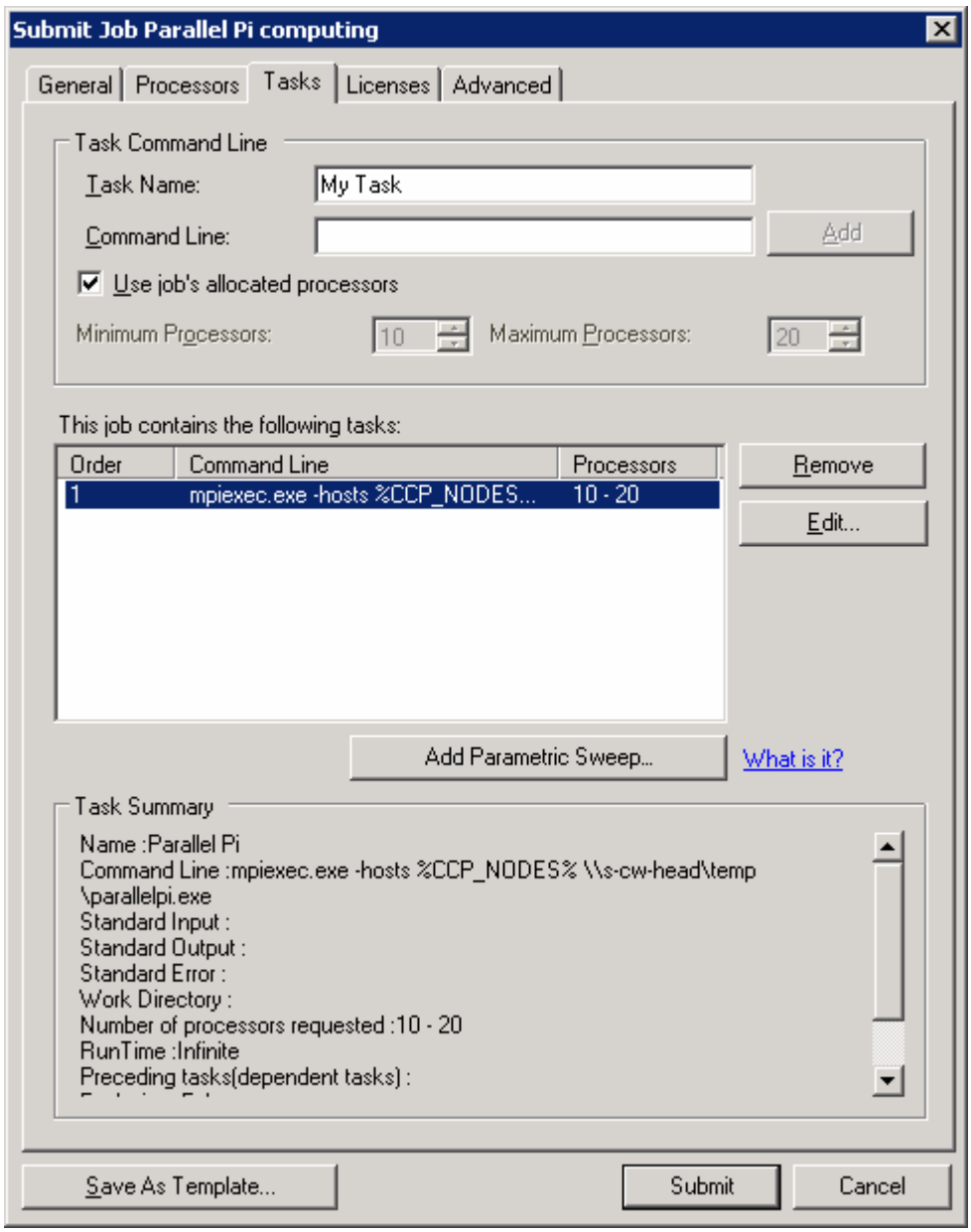
- In the window tab **Processors** enter the minimum and the maximum number of processors required for executing the job (for instance, 10 and 20 correspondingly). Go to the window tab **Tasks** to add new tasks to the job,



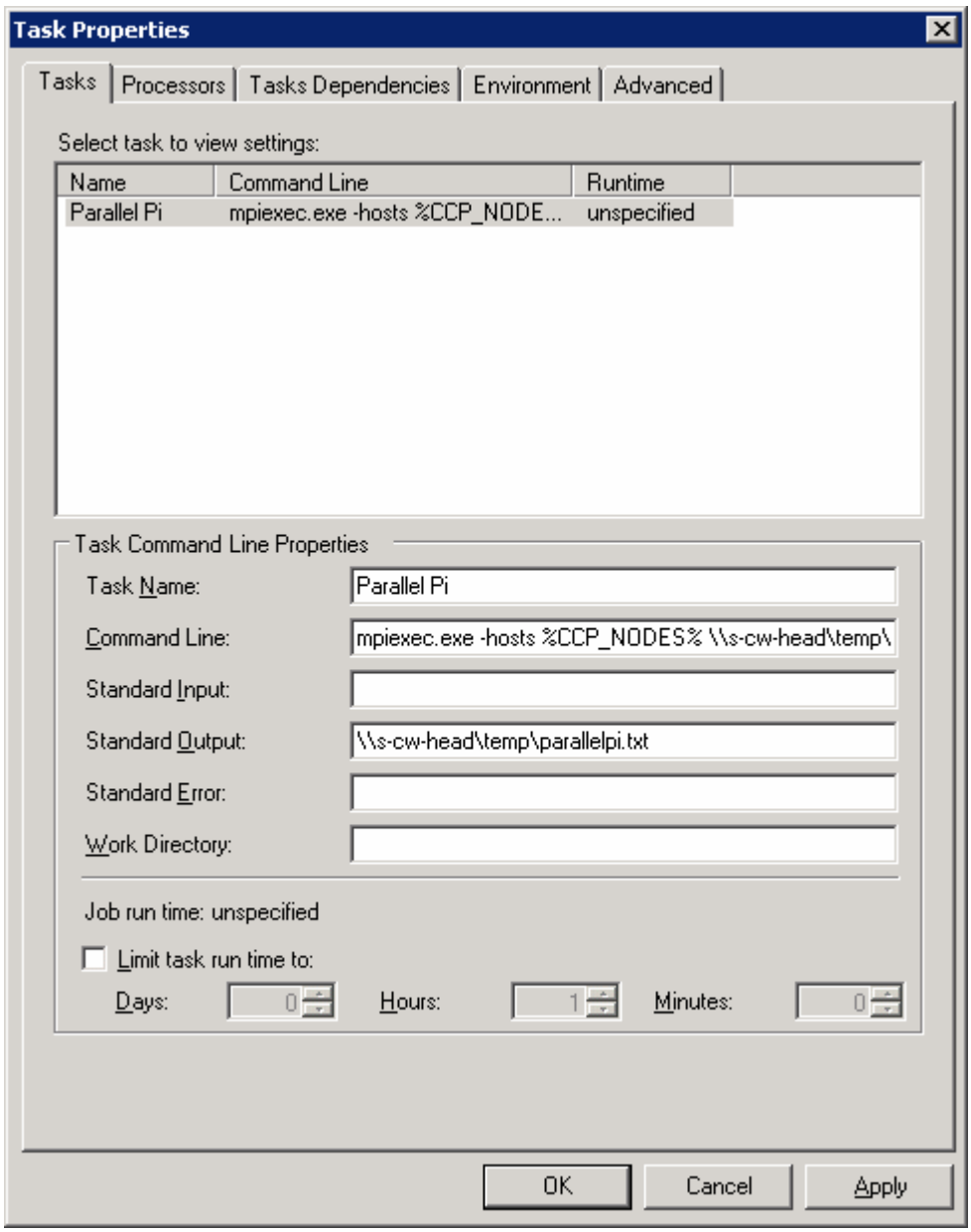
- Add the task name (the field **Task Name**) and the command to be executed (the field **Command Line**). Launching the tasks developed for MS MPI must be executed with the use of the special utility **mpiexec.exe**, which accepts the name of the parallel program, the list of nodes, where the launch will be executed, and the parameters of the program being launched, as its parameters. The list of nodes is set by the parameter "**-hosts**". If the nodes have been allocated automatically by the scheduler, the list of nodes will be contained in the environment variable **CCP_NODES**. The value of the variable should be given to the utility as a parameter. The example of the command for launching the parallel program is "**mpiexec.exe –hosts %CCP_NODES% \\s-cw-head\temp\parallelpi.exe**". Press the button **Add** to add the task to the job,



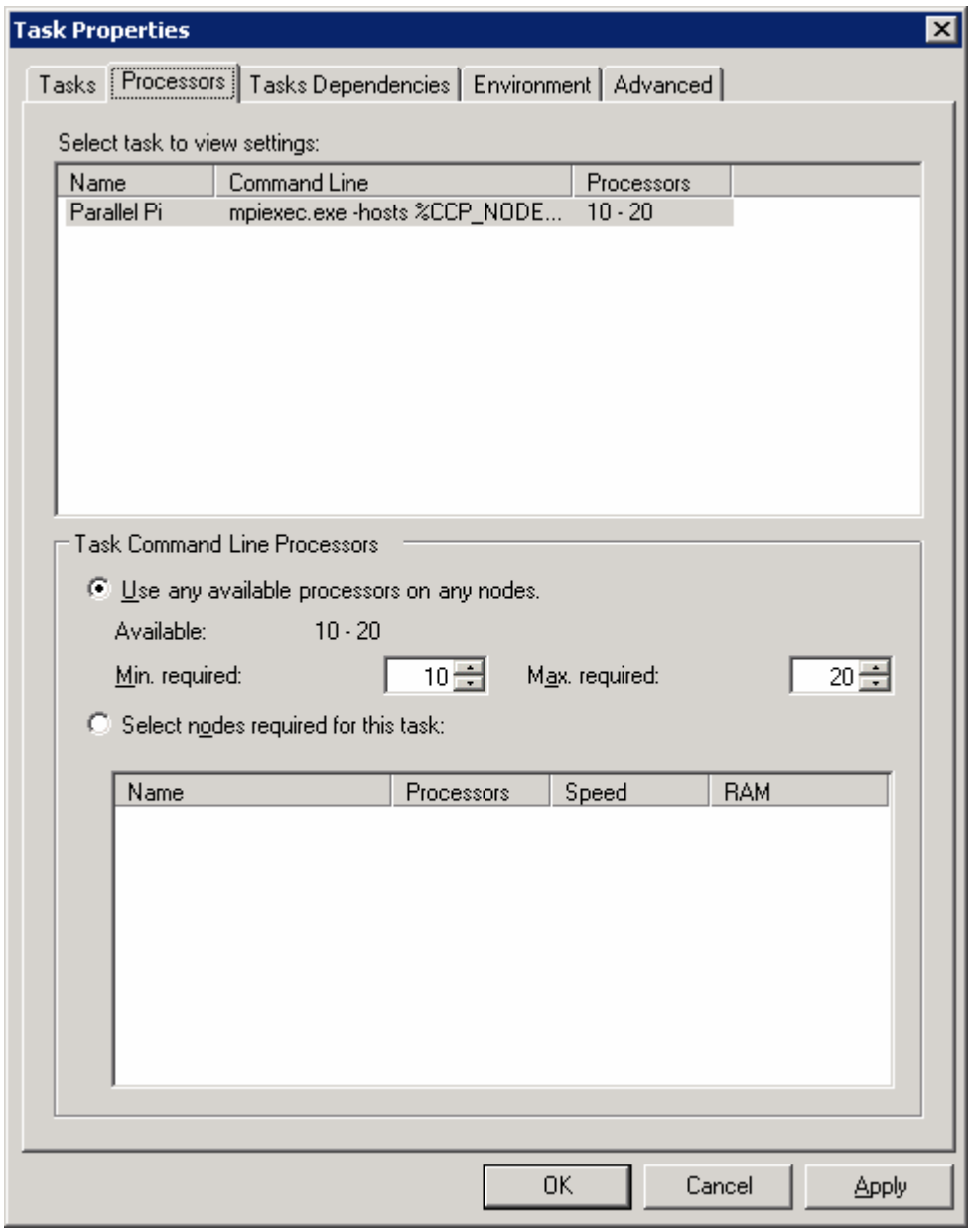
- The added task will appear in the task list of the current job (the list **This job contains the following tasks**). Select it in the list and press the button **Edit** to edit the additional task parameters,



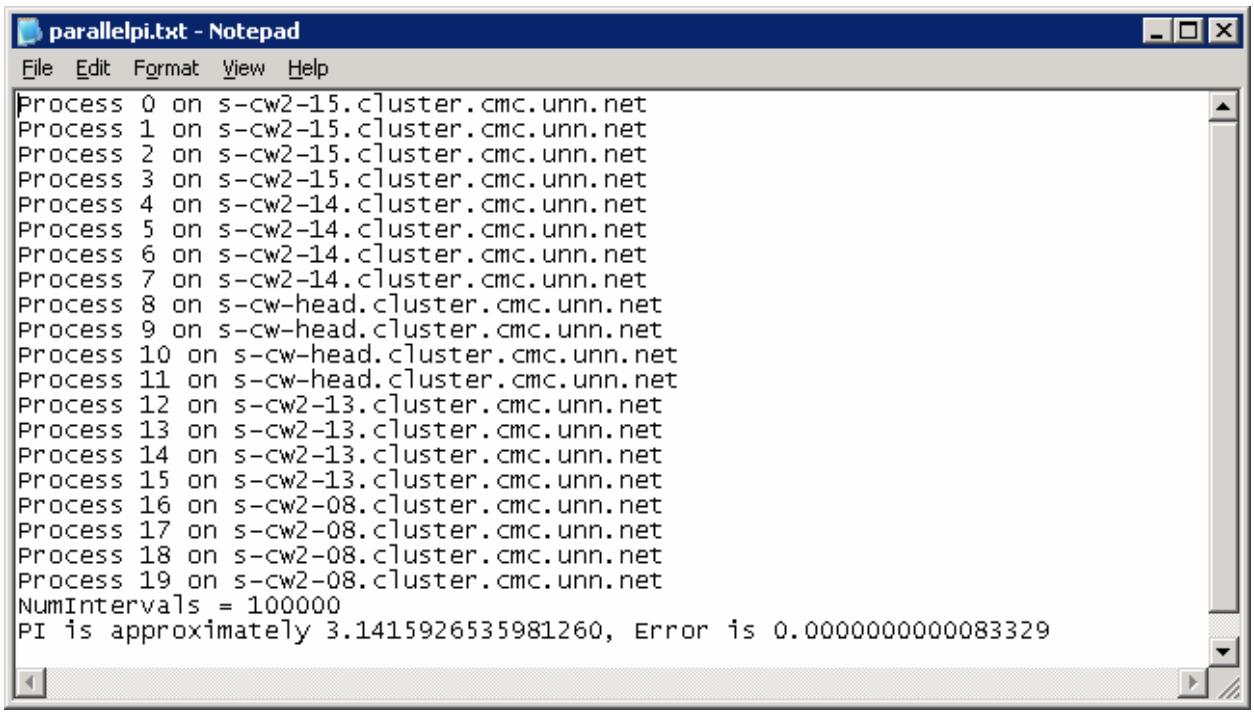
- In the new window enter the path to the file, where the standard output stream of the console application will be redirected (the field **Standard Output**). Choose the window tab **Processors**,



- In the window tab **Processors** in the upper list (**Select task to view settings**) select the task, where you want to change the settings, and give the maximum and the minimum number of processors for the selected task (the fields **Min. required** and **Max. required**). Press **OK** to save the changes to be done and return to the job settings,



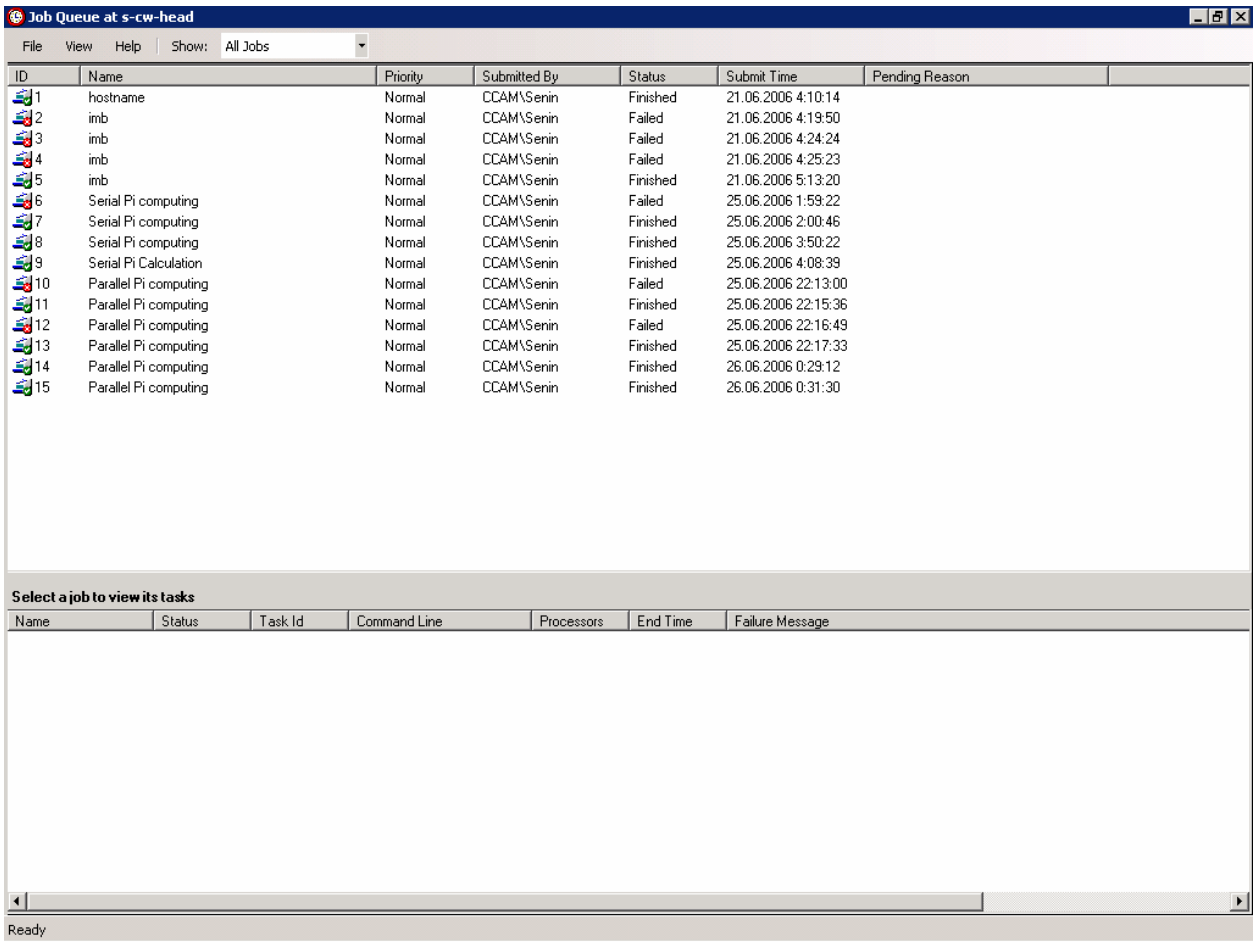
- Press the button **Submit** to add the job to the queue. In the window requesting for the password, enter the name and the password of the user, who is authorized to launch tasks on the cluster, and press **OK**. The job will appear in the queue **Job Manager**. After the job execution is completed, its status will change for **Finished**. The file, given in the task settings for redirection of the standard output stream, contains the results of the program execution,



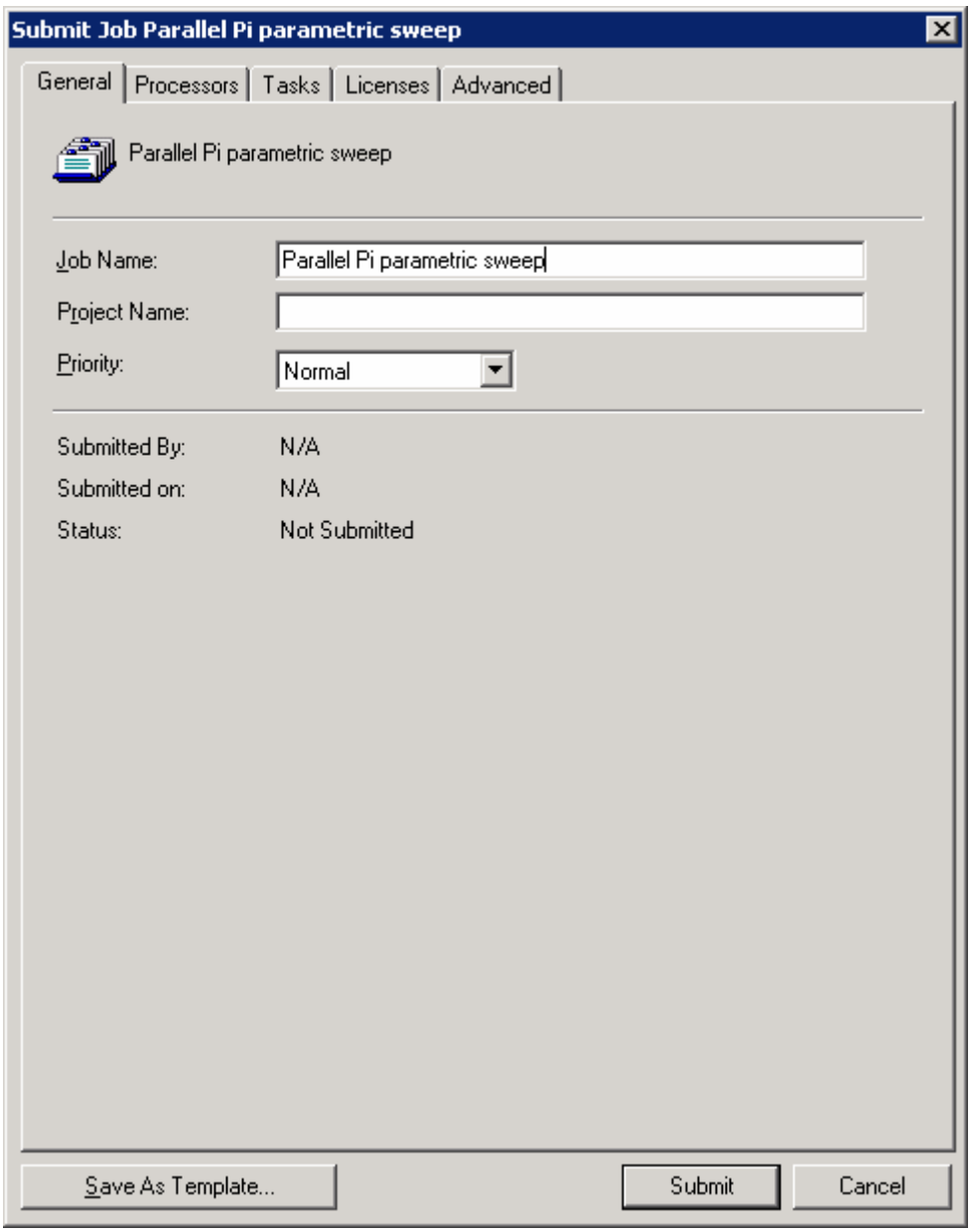
Exercise 4 – Launching a Parametric Sweep

Here we will consider the launch of parametric sweep within a job. The parametric sweep is a series of launches of the same program with different parameters. As an example you can run a series of several hundreds experiments on computing the Pi in order to study the rate of the method convergence to the solution. As an example of the program for this Exercise we will use the program of parallel computation of the Pi:

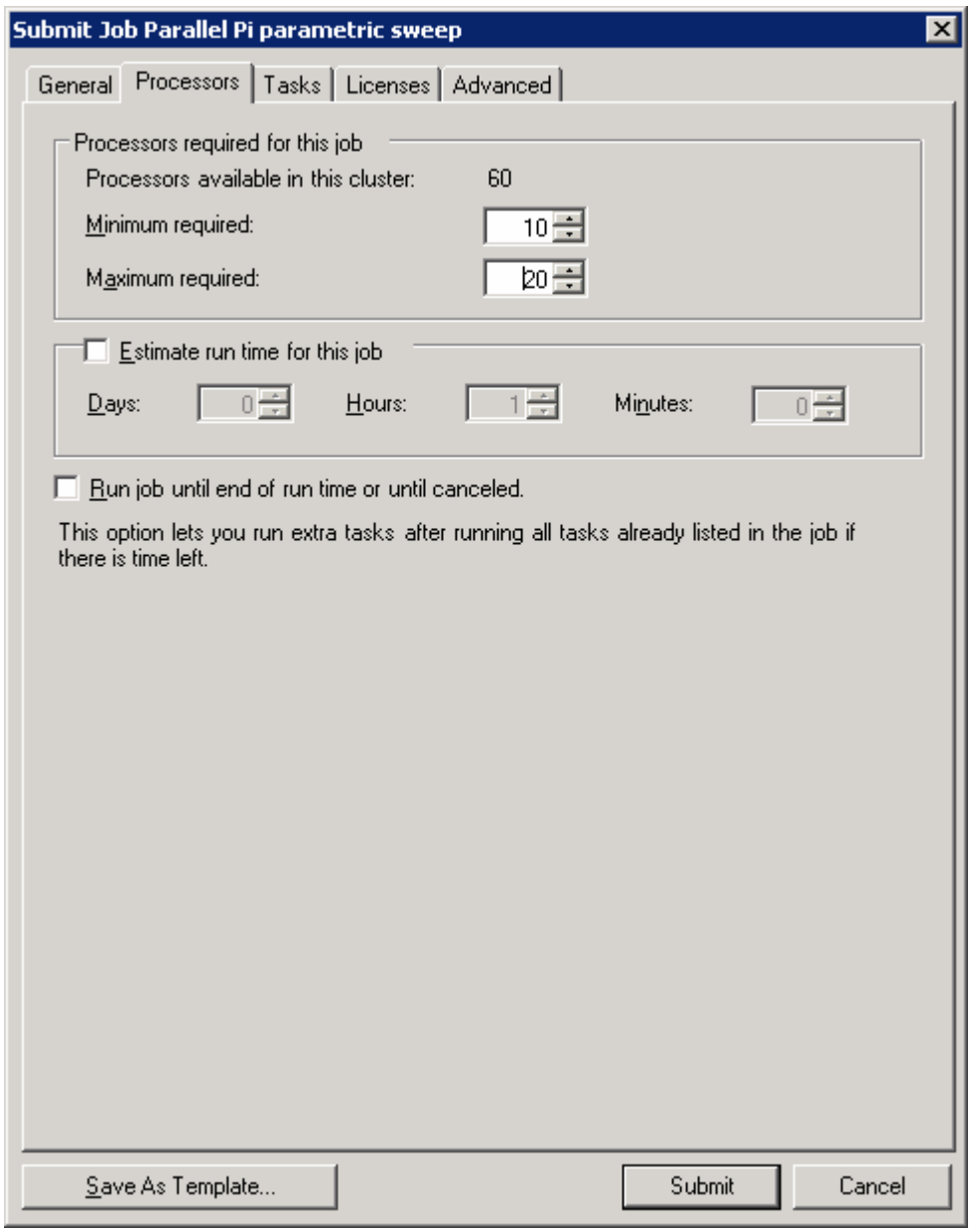
- Open **Computer Cluster Job Manager** (**Start->All Programs->Microsoft Compute Cluster Pack->Compute Cluster Job Manager**) to create the parametric sweep,



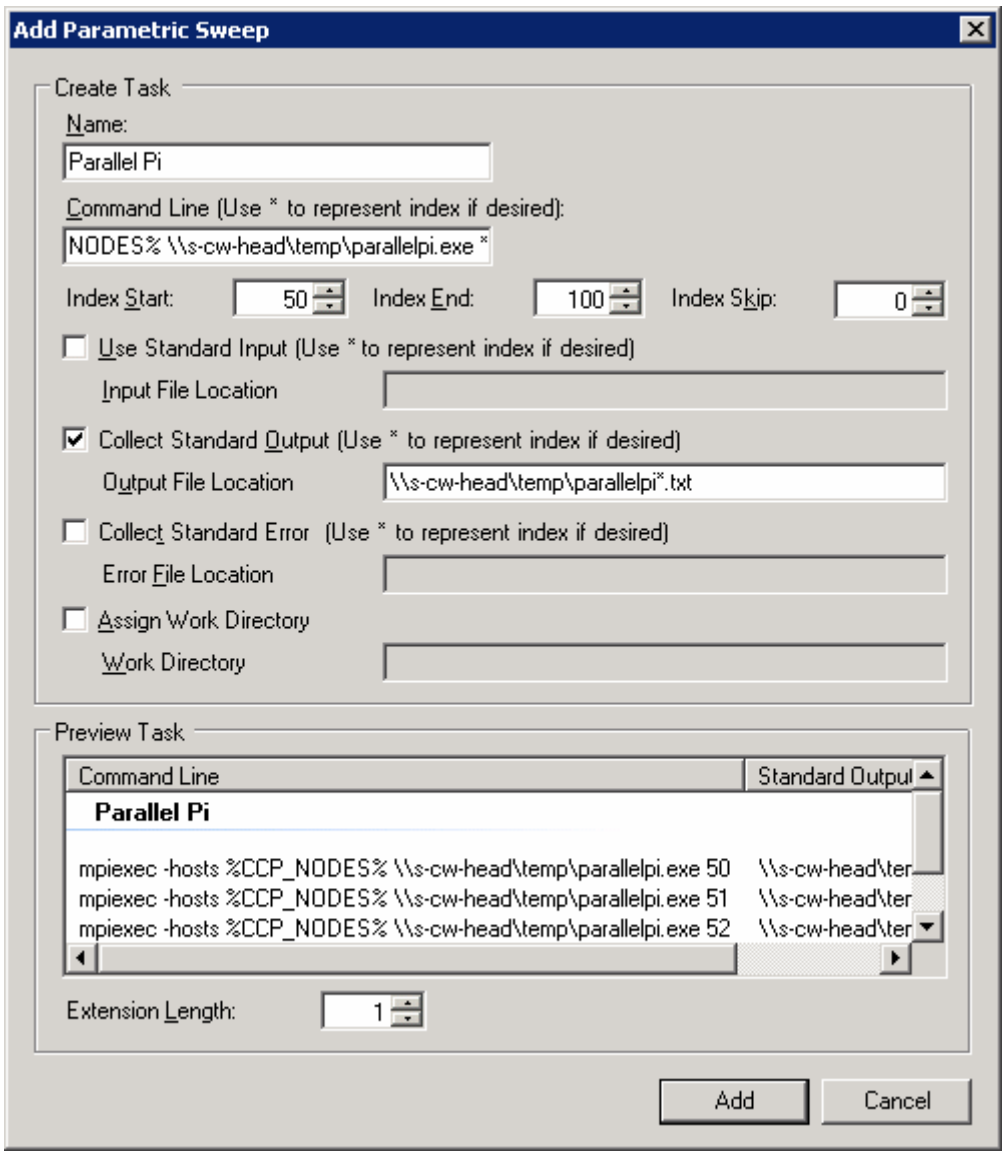
- In the window of the job manager choose the menu option **File->Submit Job** to add a new job to the queue,
- In the window of adding the job to the queue enter the job name (the field **Job Name**). Go to the window tab **Processors**,



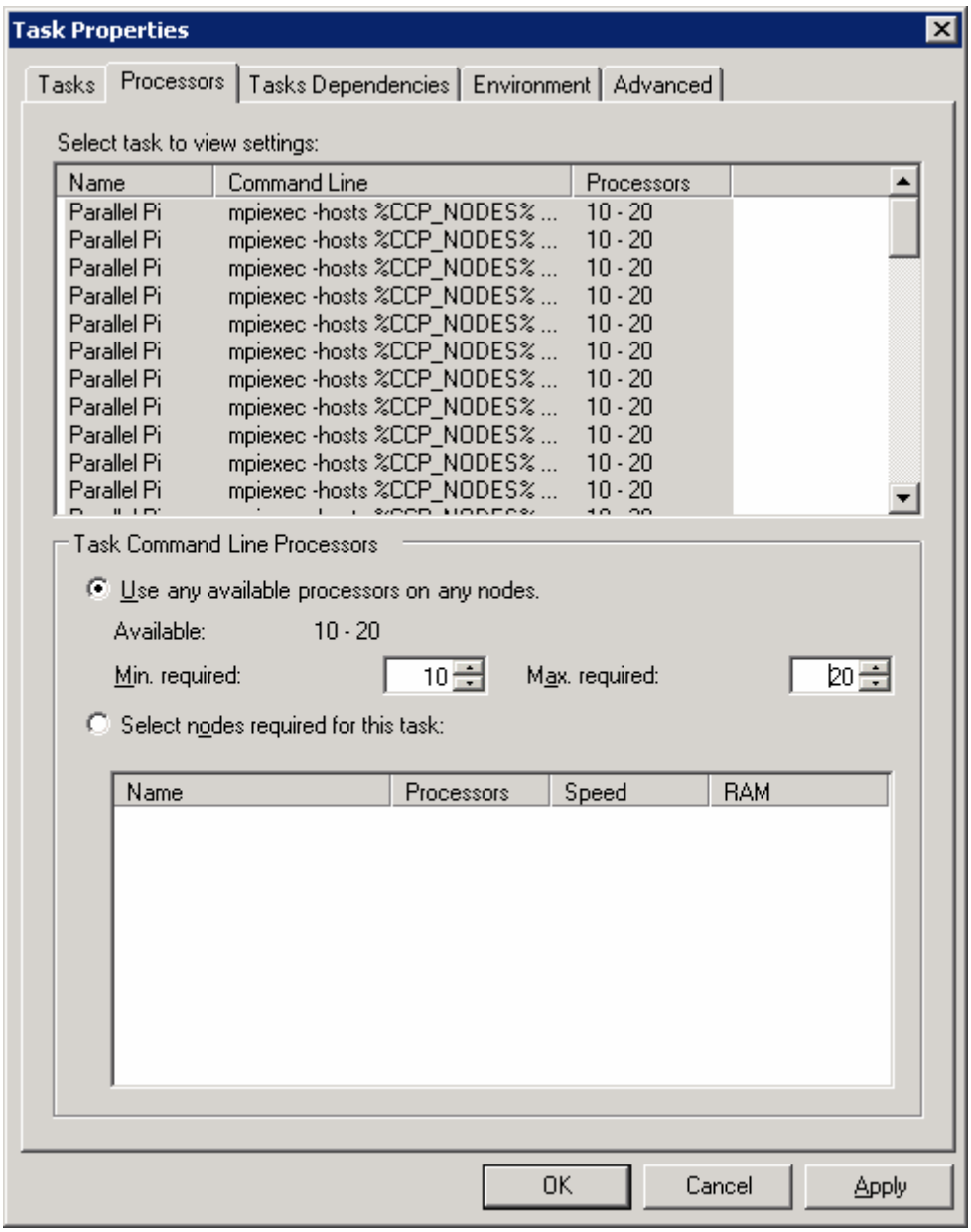
- In the window tab **Processors** enter the minimum and the maximum number of processors required for executing the job (for instance, 10 and 20 correspondingly). Go to the window tab **Tasks** and press the button **Add parametric Sweep** in the window tab to add new tasks to the job,



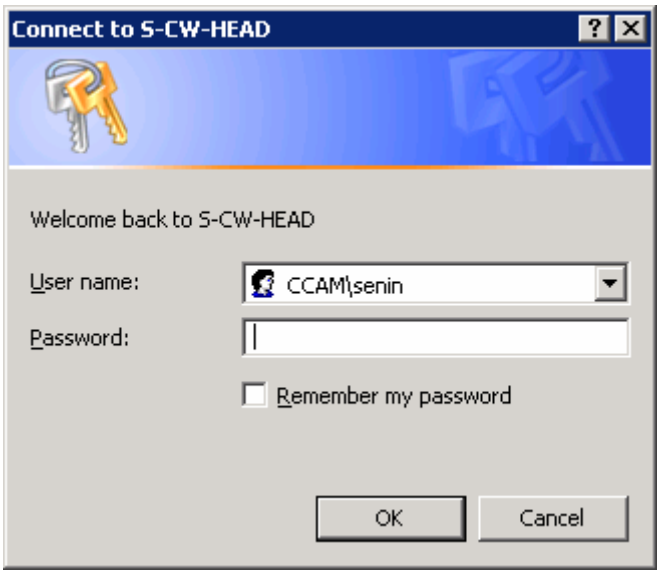
- In the window of adding the parametric sweep enter the name assigned to each new task (the field **Name**). Enter the command for the task using the asterisk (the symbol "*****") as the argument parameter of the command line. The symbol "*****" for each particular command will be replaced by an integer number, the range of change for the number will be specified in the fields **Index Start** and **Index End**. The index for our task (the number of intervals of the numerical integration) may change, for instance, from 50 to 100. Thus, the command may be the following: "**mpiexec.exe –hosts %CCP_NODES% \\s-cw-head\temp\parallelpi.exe ***". Specify the files, where the standard output stream will be redirected, using "*****" as a parameter. For instance: "**\\s-cw-head\temp\parallelpi*.txt**". Press **OK** to add a task sweep to the job,



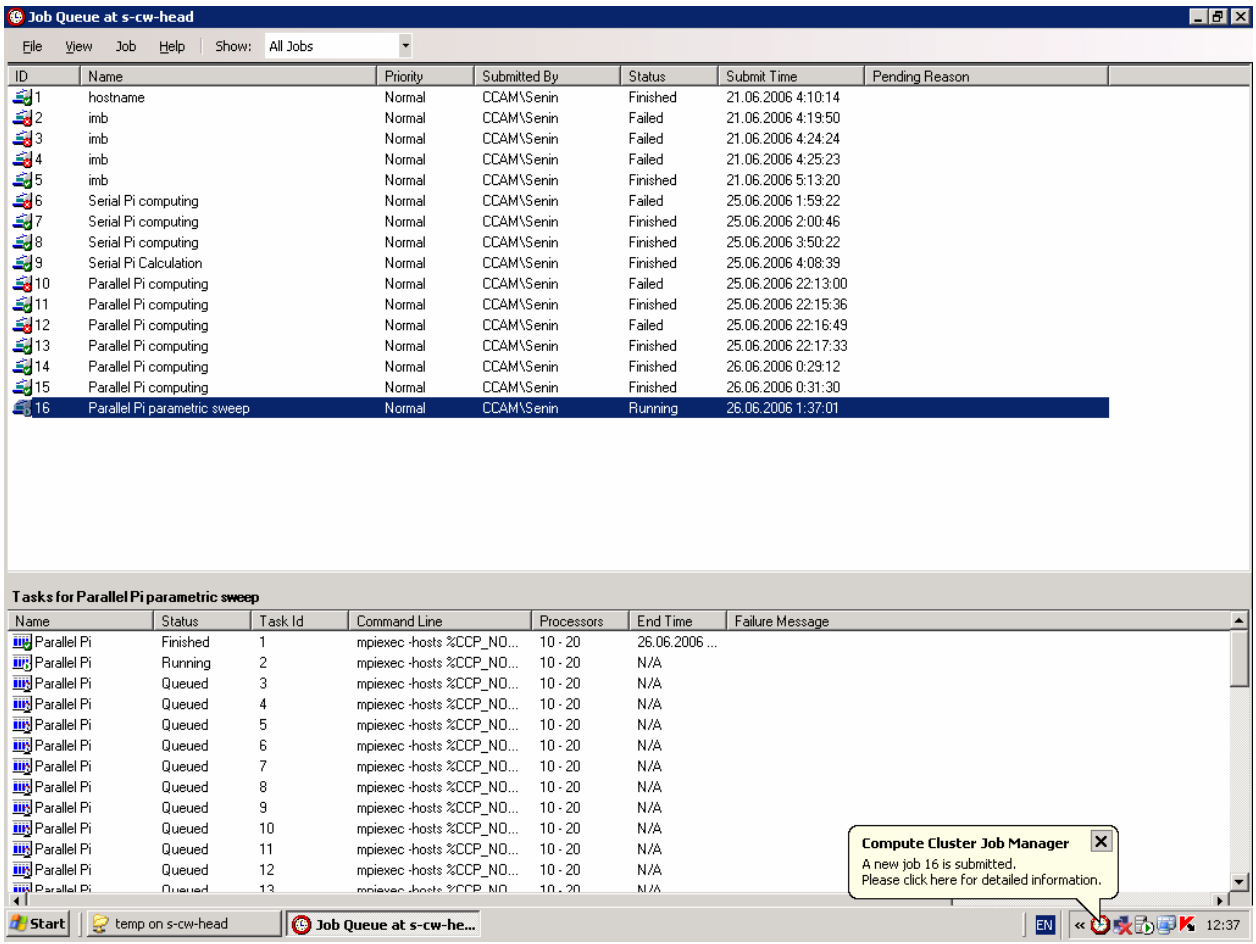
- In the window of the job setting select all the tasks contained in the job (in order to select several tasks, use the key **Shift**), press the button **Edit** to specify the number of the processors required for the tasks. In the new window go to the window tab **Processors**, select the option **Use any available processors on any nodes** and specify, for instance, 10 as the minimum number of processors, and 20 as the maximum one. Press **OK**. In the open window press the button **Submit** to add the job to the queue,



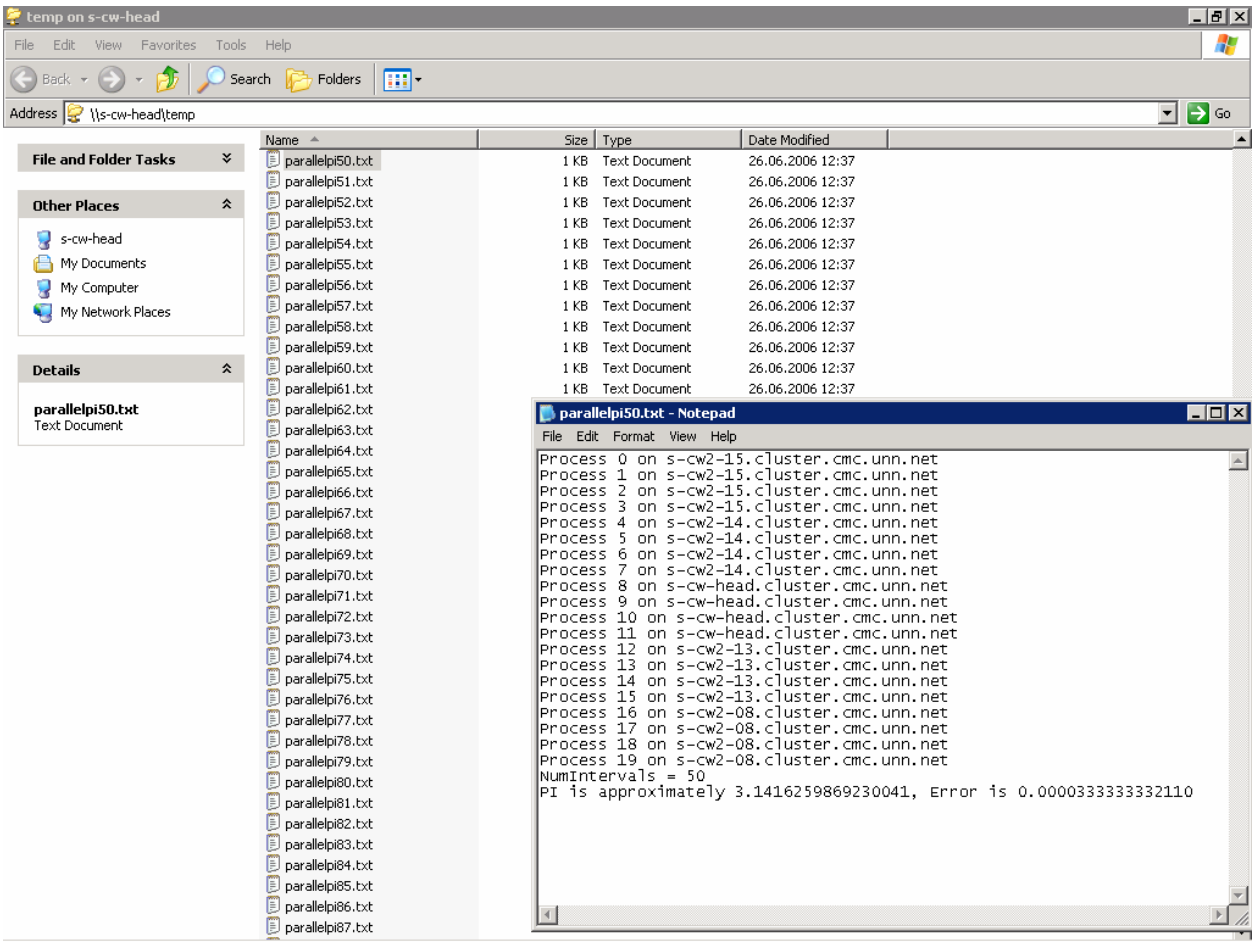
- Enter the name and the password of the user authorized to run tasks on the cluster and press **OK**,



- A new job will appear in the window **Job Manager.** If you select it, you will be able to track the execution of its tasks in the lower list. When the job is completed, its status will change for **Finished**,



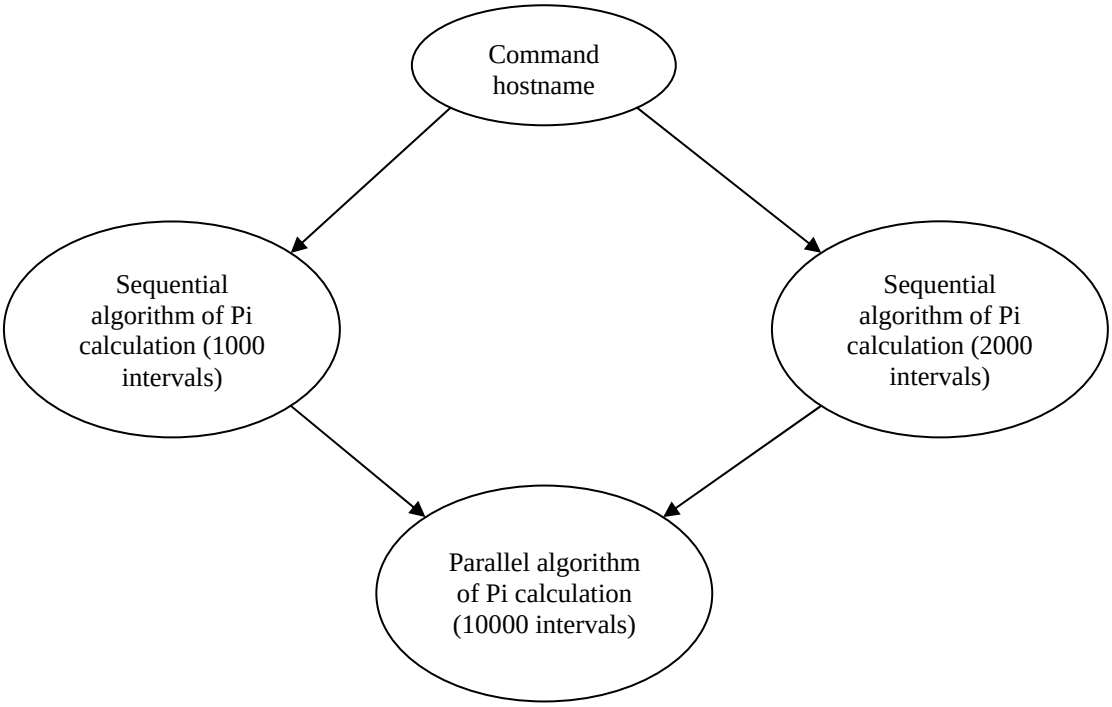
- You can look through the results of the job execution in the files, which you have specified for saving the redirected output stream.



Exercise 5 – Launching a Work Flow

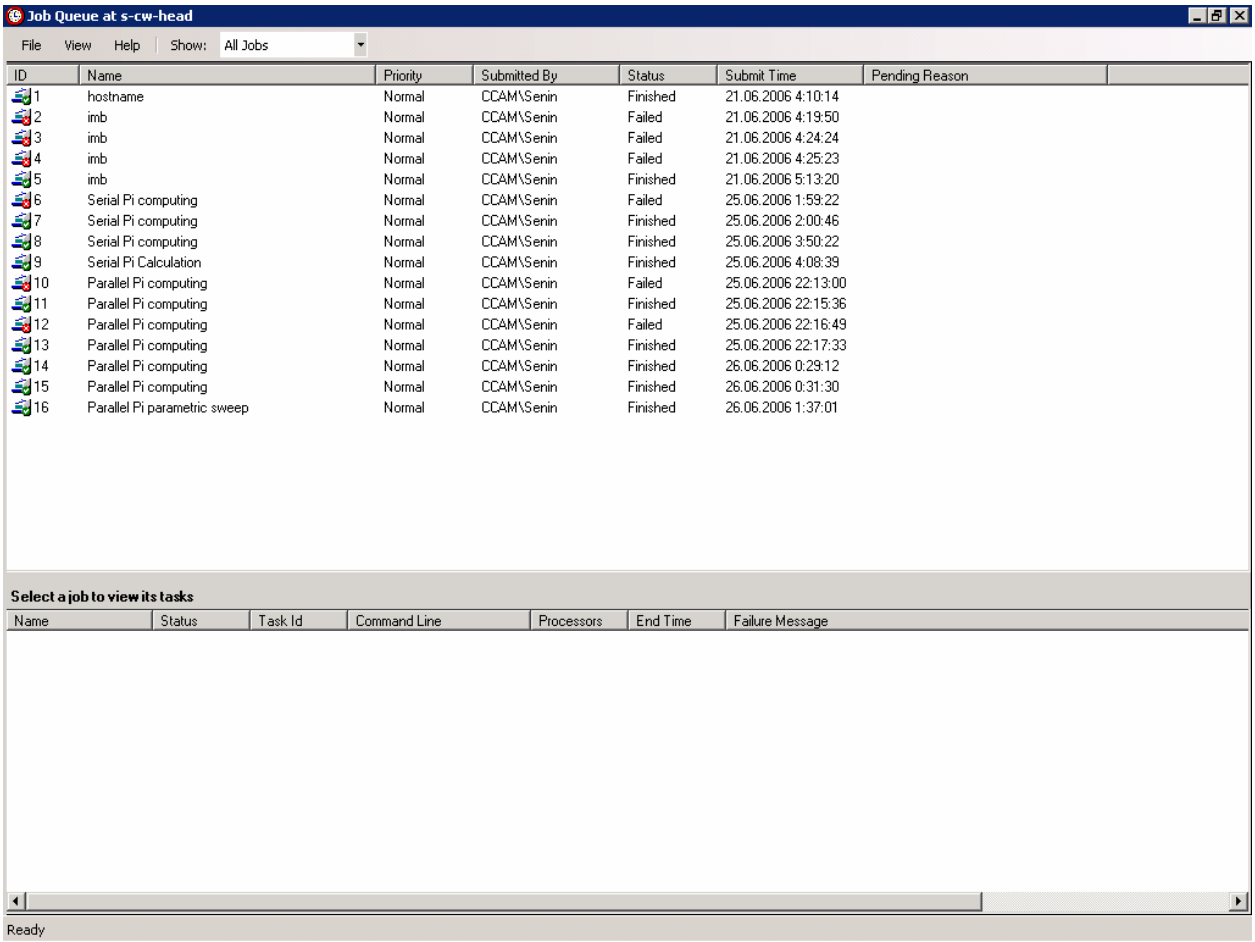
The work flow is used if the execution of a certain task within a job requires the results of the other task execution, which creates requirements to the task execution sequence. These requirements are convenient to set in the form of the acyclic oriented graph, where each vertex is a task, and the arrow shows the dependence of the vertex - child against the vertex - parent. In this case the task execution sequence is defined by the following simple rule: neither of the tasks can be launched until all the tasks, which correspond to its parents on the dependence graph, are executed.

We can consider the following task dependence graph as an example:



Let us set this dependence graph in **CCS 2003**:

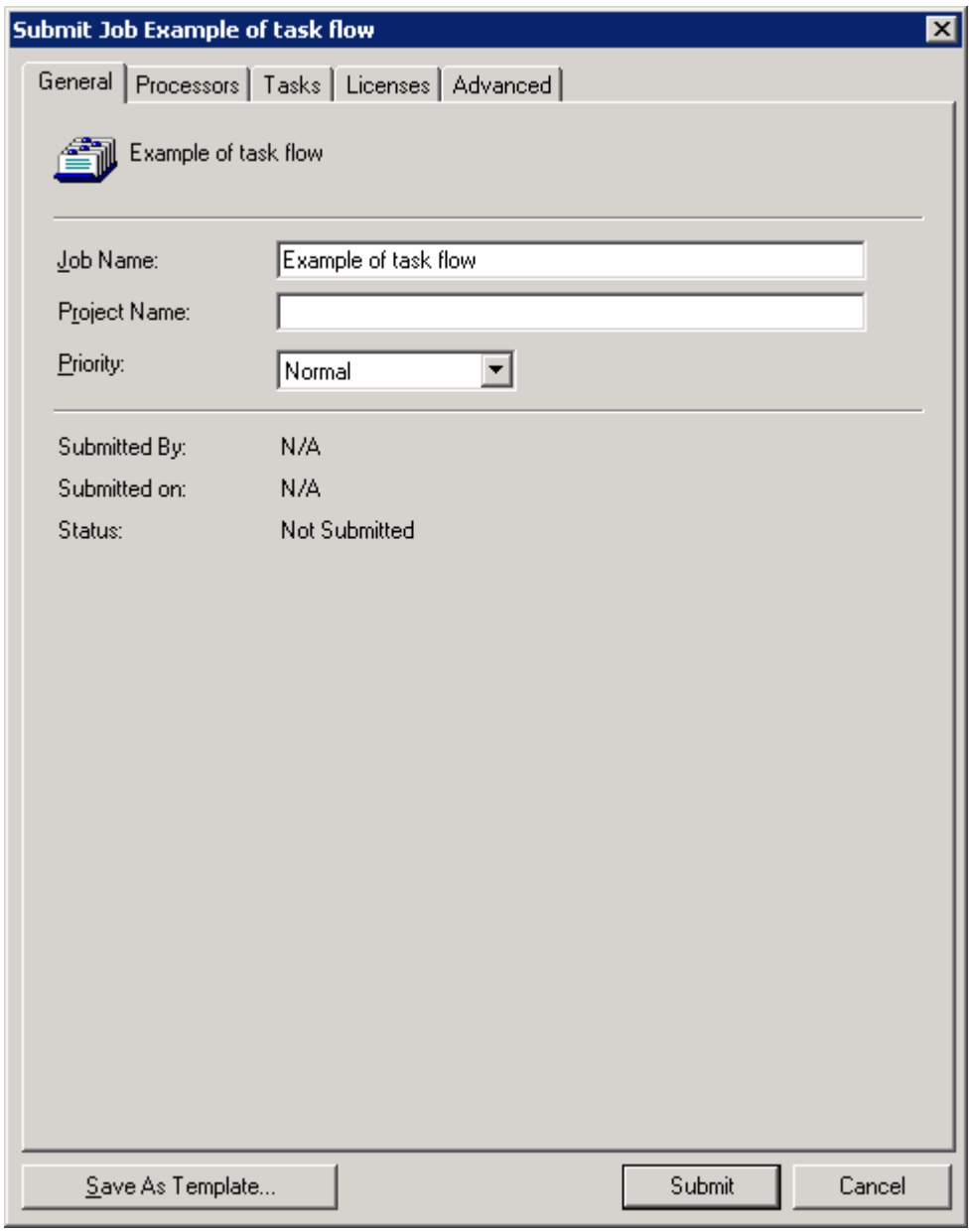
- Open **Computer Cluster Job Manager** (**Start->All Programs->Microsoft Compute Cluster Pack->Compute Cluster Job Manager**),



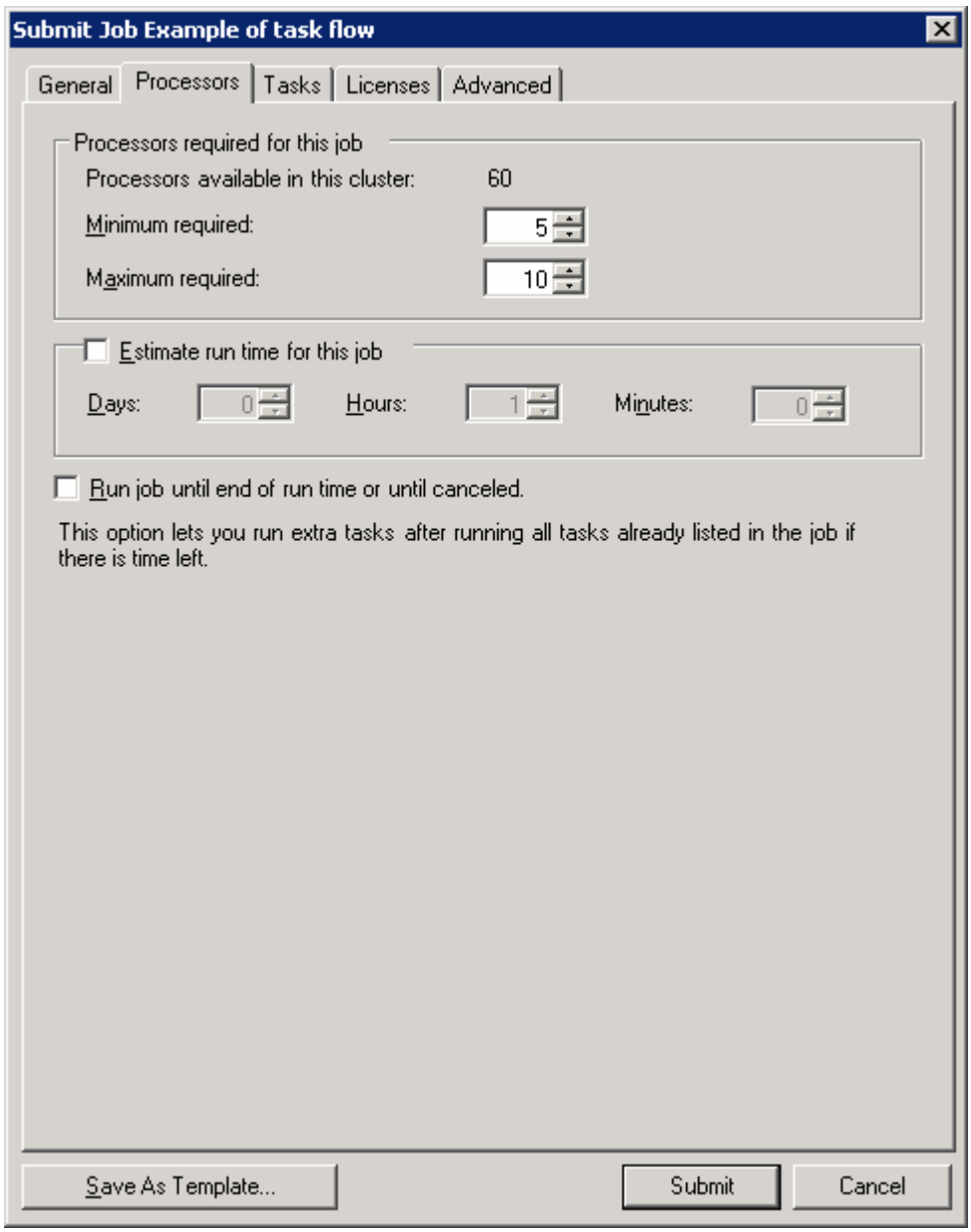
ID	Name	Priority	Submitted By	Status	Submit Time	Pending Reason
1	hostname	Normal	CCAM\Senin	Finished	21.06.2006 4:10:14	
2	imb	Normal	CCAM\Senin	Failed	21.06.2006 4:19:50	
3	imb	Normal	CCAM\Senin	Failed	21.06.2006 4:24:24	
4	imb	Normal	CCAM\Senin	Failed	21.06.2006 4:25:23	
5	imb	Normal	CCAM\Senin	Finished	21.06.2006 5:13:20	
6	Serial Pi computing	Normal	CCAM\Senin	Failed	25.06.2006 1:59:22	
7	Serial Pi computing	Normal	CCAM\Senin	Finished	25.06.2006 2:00:46	
8	Serial Pi computing	Normal	CCAM\Senin	Finished	25.06.2006 3:50:22	
9	Serial Pi Calculation	Normal	CCAM\Senin	Finished	25.06.2006 4:08:39	
10	Parallel Pi computing	Normal	CCAM\Senin	Failed	25.06.2006 22:13:00	
11	Parallel Pi computing	Normal	CCAM\Senin	Finished	25.06.2006 22:15:36	
12	Parallel Pi computing	Normal	CCAM\Senin	Failed	25.06.2006 22:16:49	
13	Parallel Pi computing	Normal	CCAM\Senin	Finished	25.06.2006 22:17:33	
14	Parallel Pi computing	Normal	CCAM\Senin	Finished	26.06.2006 0:29:12	
15	Parallel Pi computing	Normal	CCAM\Senin	Finished	26.06.2006 0:31:30	
16	Parallel Pi parametric sweep	Normal	CCAM\Senin	Finished	26.06.2006 1:37:01	

Name	Status	Task Id	Command Line	Processors	End Time	Failure Message

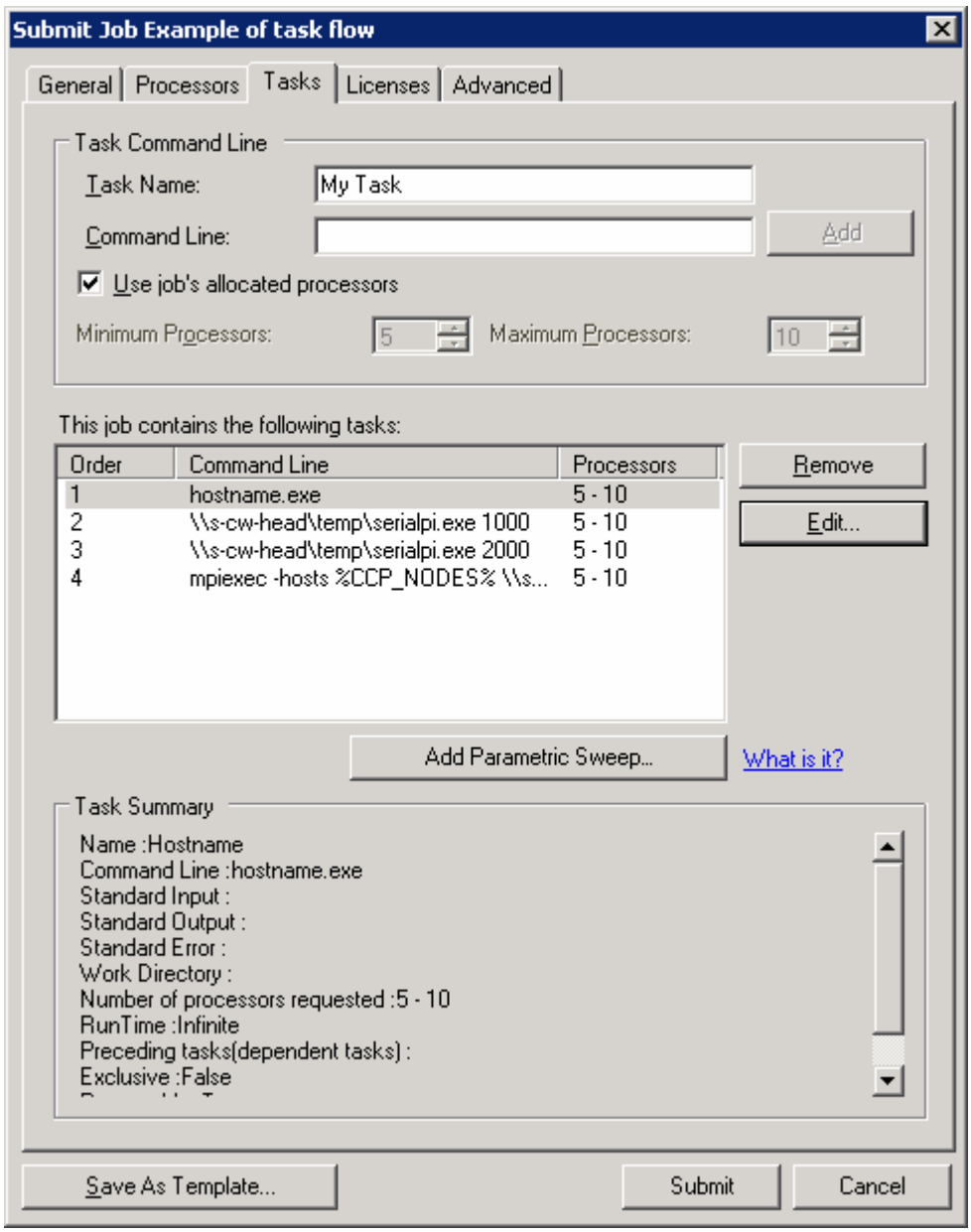
- In the new window of job manager choose the menu option **File->Submit Job**,
- In the window of adding a job to the queue enter the name of the job (the field **Job Name**). Go to the window tab **Processors**,



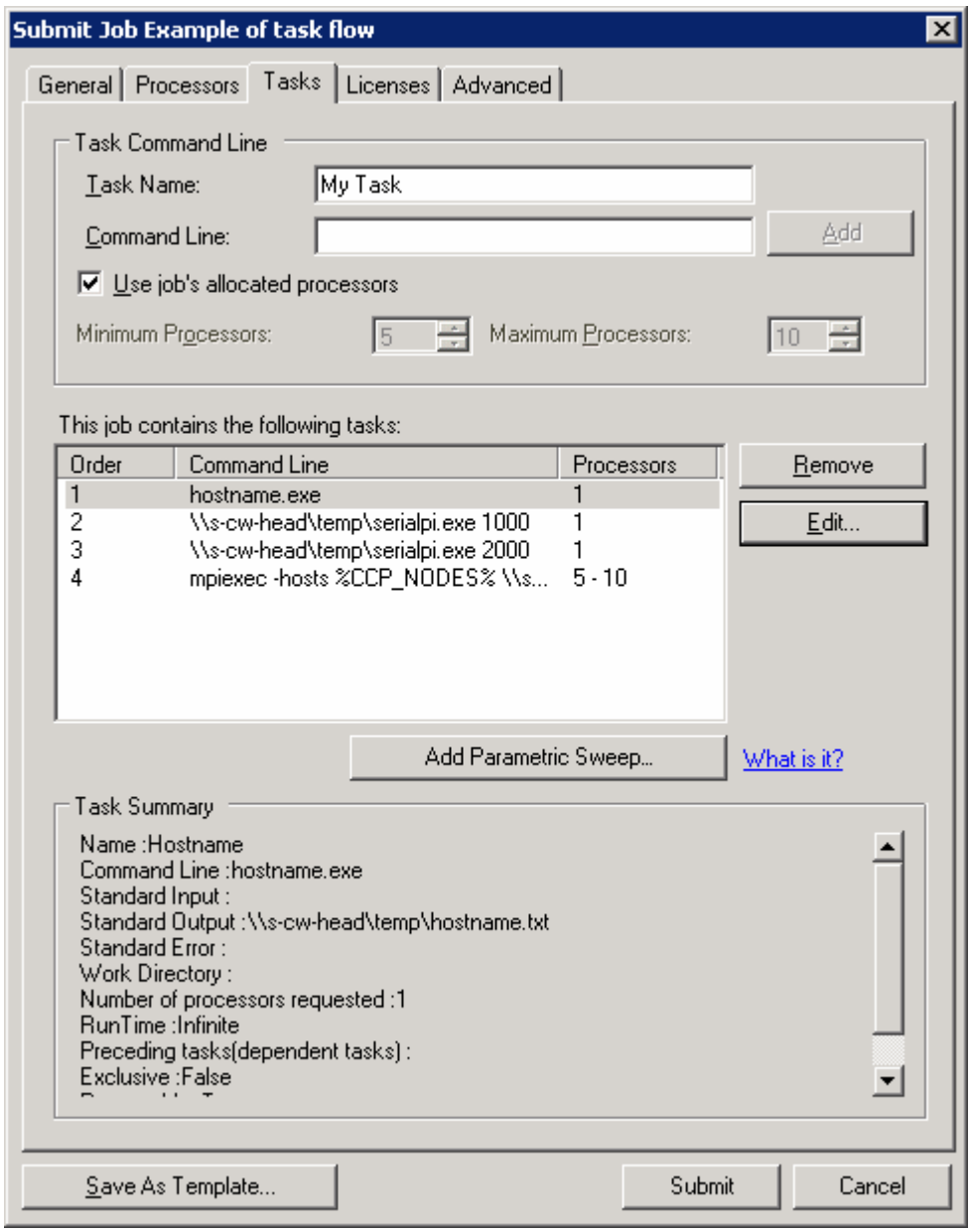
- In the window tab **Processors** enter the minimum and the maximum number of processors required for executing the job (for instance, 5 and 10 correspondingly). Go to the window tab **Tasks** to add new tasks to the job,



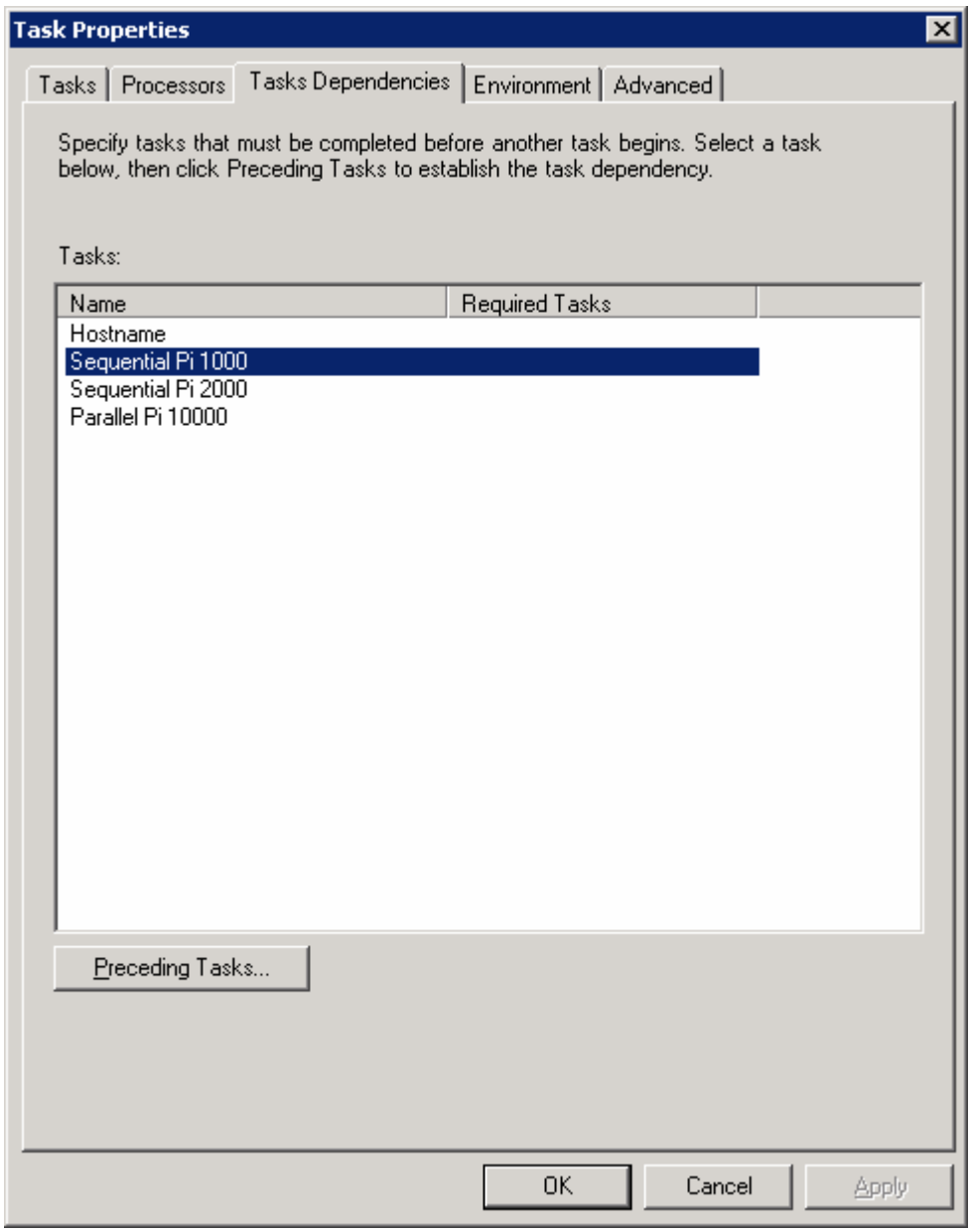
- Add the following 4 tasks to the job sequentially:
  - The task named "**Hostname**" with the command "**hostname.exe**",
  - The task named "**Serial Pi 1000**" with the command "**\\s-cw-head\temp\serialpi.exe 1000**" (change the path to the executed file of the program for the existing one),
  - The task named "**Serial Pi 2000**" with the command "**\\s-cw-head\temp\serialpi.exe 2000**" (change the path to the executed file of the program for the existing one),
  - The task named "**Parallel Pi 10000**" with the command "**mpiexec –hosts %CCP_NODES% \\s-cw-head\temp\parallelpi.exe 10000**" (change the path to the executed file of the program for the existing one),



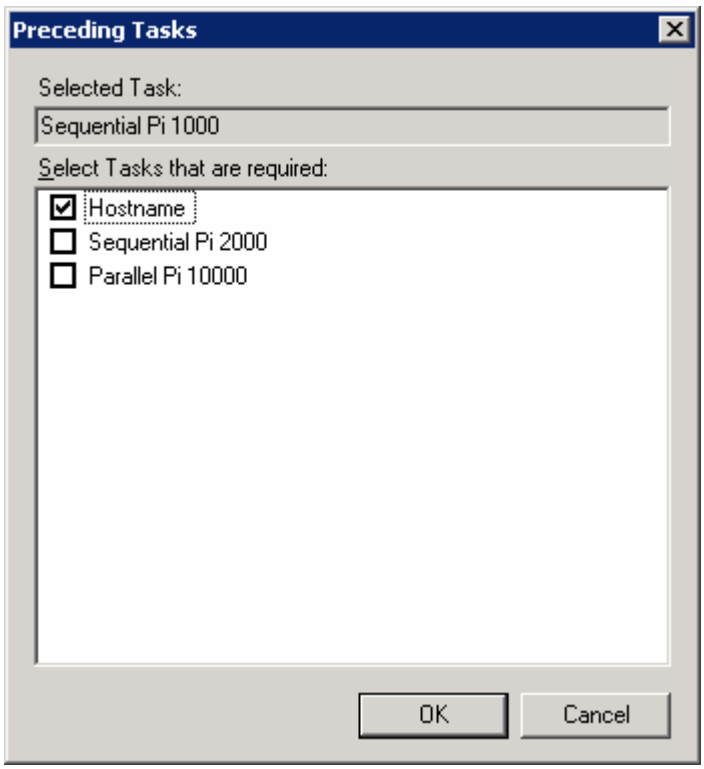
- Set the additional parameters of the tasks:
  - For the tasks "**Hostname**", "**Serial Pi 1000**" and "**Serial Pi 2000**" set the maximum required number of processors as 1, set the file for redirecting the standard output stream for each of the 3 tasks,
  - For the task "**Parallel Pi 10000**" set the minimum and the maximum required numbers of processors as 5 and 10 correspondingly, set the file for redirecting the standard output stream,
- Go to the window tab **Tasks Dependencies** of the task properties (to go to the task properties, press the button **Edit** in the window tab **Tasks** of the window **Submit Job**),



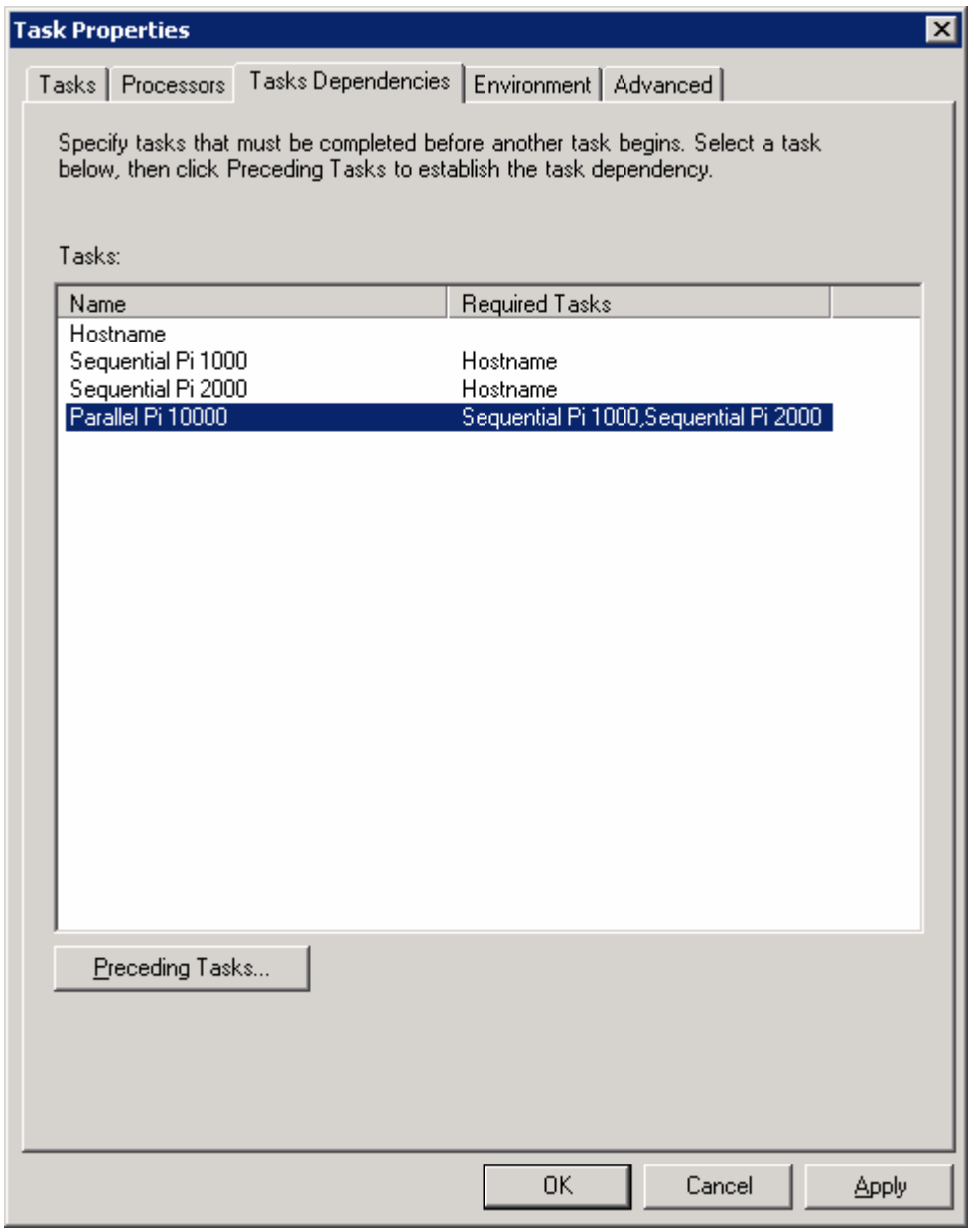
- Select the task "**Serial Pi 1000**" and press the button **Preceding Tasks** to set the tasks, from which the considered task depends on,



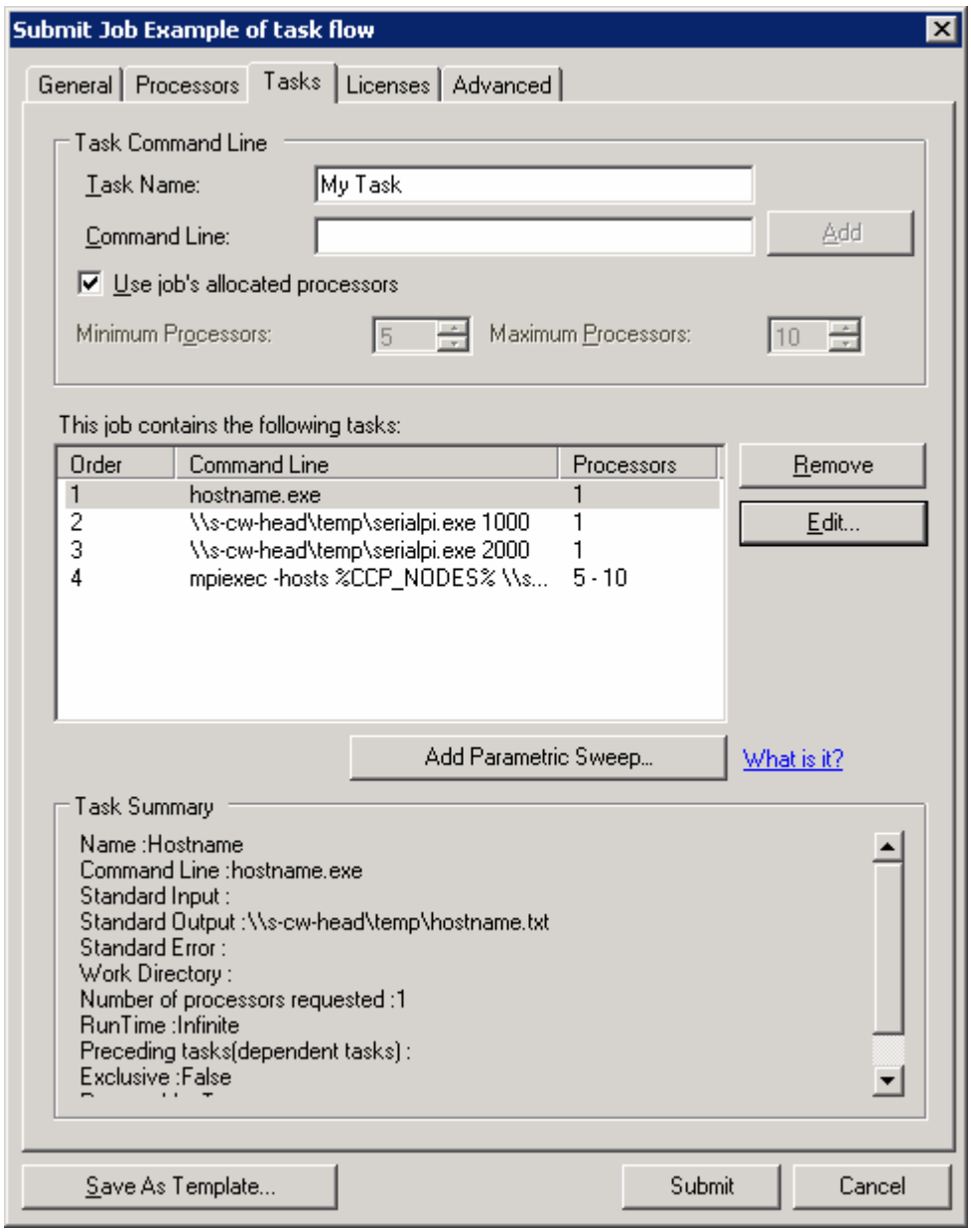
- In the new window tick the task "**Hostname**". Press **OK**,



- Set the dependence against "**Hostname**" for the task "**Serial Pi 2000**". Set the dependence against the tasks "**Serial Pi 1000**" and "**Serial Pi 2000**" for the task "**Parallel Pi 10000**". Press **OK** to change the changes to be done,



- In the window **Submit Job** press the button **Submit** to add the job to the queue,



- Enter the name and the password of the user authorized to run jobs on the cluster,
- The scheduler CCS 2003 will first start the task "**Hostname**", then the tasks "**Serial Pi 1000**" and "**Serial Pi 2000**" in parallel, and only after that it will start the task "**Parallel Pi 10000**".

Optional Exercise. Evaluating the Network Performance Parameters

The necessity to take into account not only the characteristics of individual computers (first of all the processor performance and the memory rate), but also the performance parameters of the network transmitting the data among them, has to be in the centre of attention in the process of efficient parallel program development for cluster systems. These parameters are often used for formulating the theoretical estimations of the algorithm execution time. It makes possible to predict the program execution time depending on the transmitted data size. Obtaining the network performance parameters is a separate problem, which is solved by means of launching special test programs on the particular available equipment. It is necessary to carry out these tests on each particular cluster because the data provided by the hardware vendor may vary to a great extent depending on the available software and cluster settings.

General Network Performance Parameters

The basic parameters, which are widely applied to characterize the network performance, are the latency and the bandwidth. **The latency** (the delay) is the time, spent by the hardware and the software for processing the request of sending a network message, i.e. this is the interval of time from the moment when the command to transmit data is entered to the beginning of the data transmission. Usually the latency is given in microseconds.

The network bandwidth is the maximum amount of data that can be transmitted by the network channels at a time unit. Usually it is measured in Mbyte/sec or Mbit/sec.

Methods for Evaluating the Network Performance Parameters

The basic idea of the algorithm for determining the network performance parameters, which is used in the tests of given Lab, consists in sequential transmitting messages of various lengths between two nodes, using the functions of the installed MPI implementation, and measuring the time spent on the transmission. If this data is available, the bandwidth may be determined by dividing the length of the transmitted message by the time spent on the transmission. In order to minimize the error, the transmission is repeated several times and the result is averaged. In this case the estimation of the bandwidth usually increases with the increase of the message length going to some maximum value. Usually theis maximum value (or the value obtained when a large message is transmitted) is used as the estimation of the bandwidth.

The time spent on transmitting messages of zero length is usually considered to be the latency.

This Lab describes two test programs: **Intel MPI Benchmark (IMB)** and tests developed in **the Research Computational Center of the Moscow State University (RCC MSU).**

Compiling the Benchmark Program

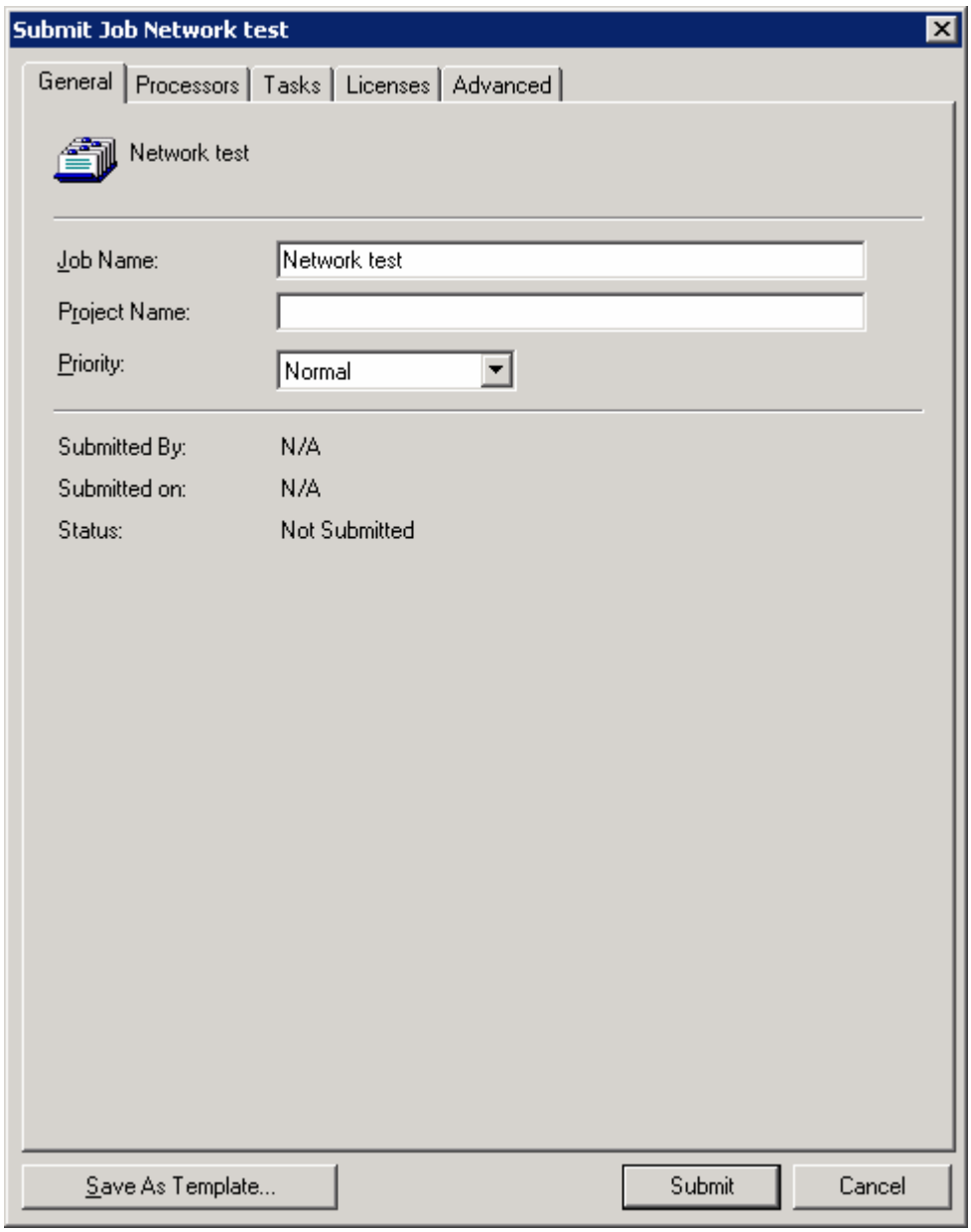
You can download the latest version of the IMB test package included into Intel Cluster Tools from Intel site http://www.intel.com/cd/software/products/asmo-na/eng/cluster/mpi/219848.htm. In order to compile IMB for Microsoft Windows, you will have to create a project in Microsoft Visual Studio 2005 on your own, by the analogy to the projects, which we described in this Lab. You can also use the project framework included into this Lab (the folder **IMB_2_3**).

Go to http://parallel.ru/ftp/tests/mpi-bench-suite.zip to download the tests developed by the **RCC MSU**. In order to compile the tests for Microsoft Windows, you will have to create a project in Microsoft Visual Studio 2005 on your own. You can also use the project framework included into this Lab (the folder **MGU_tests**).

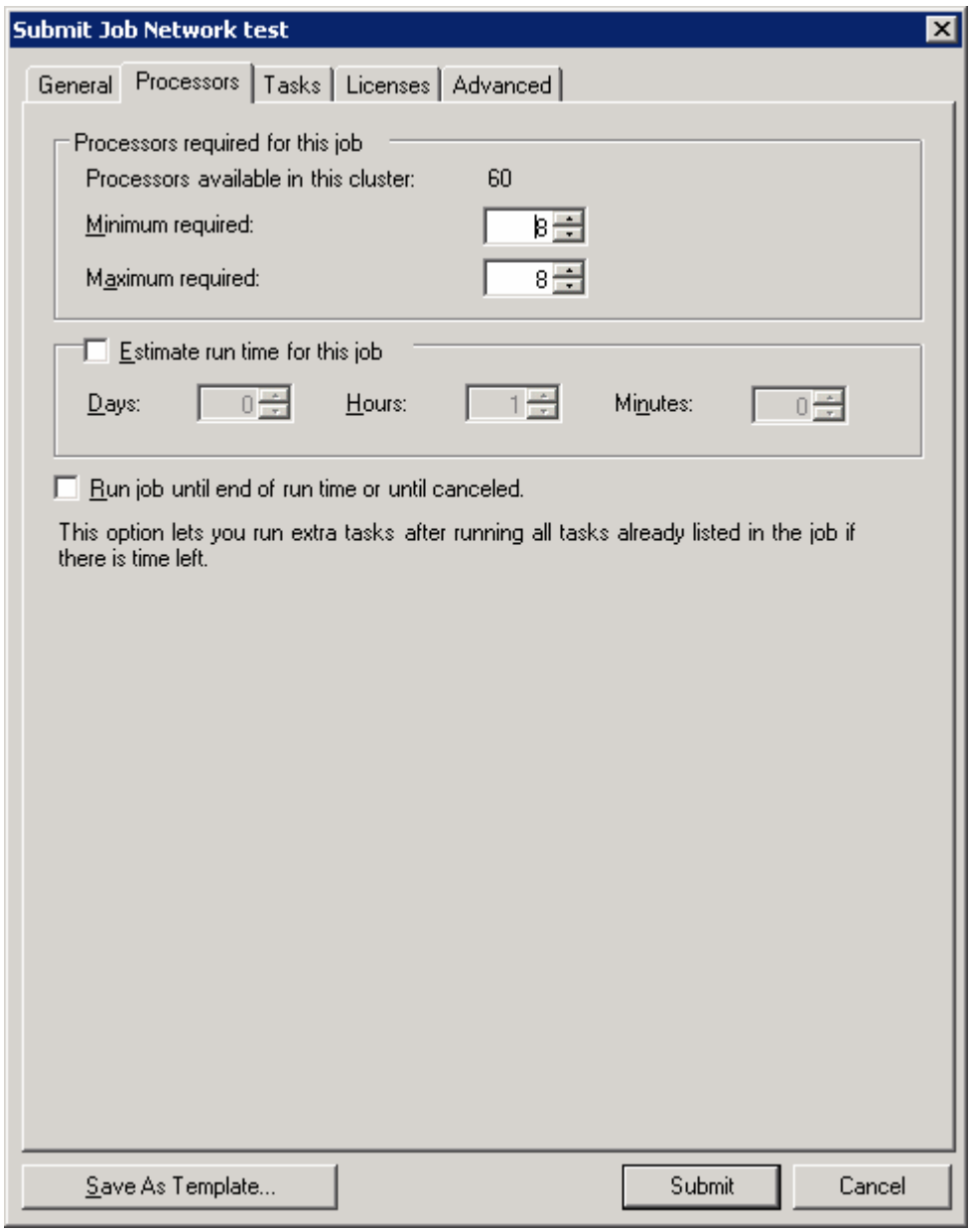
Running the Benchmarks

The benchmarks should be run on 2 network nodes (one process on each node). Thus, to launch the benchmarks in CCS 2003 it is necessary to specify the total number of processors on 2 network nodes as the job requirements and to choose these nodes manually:

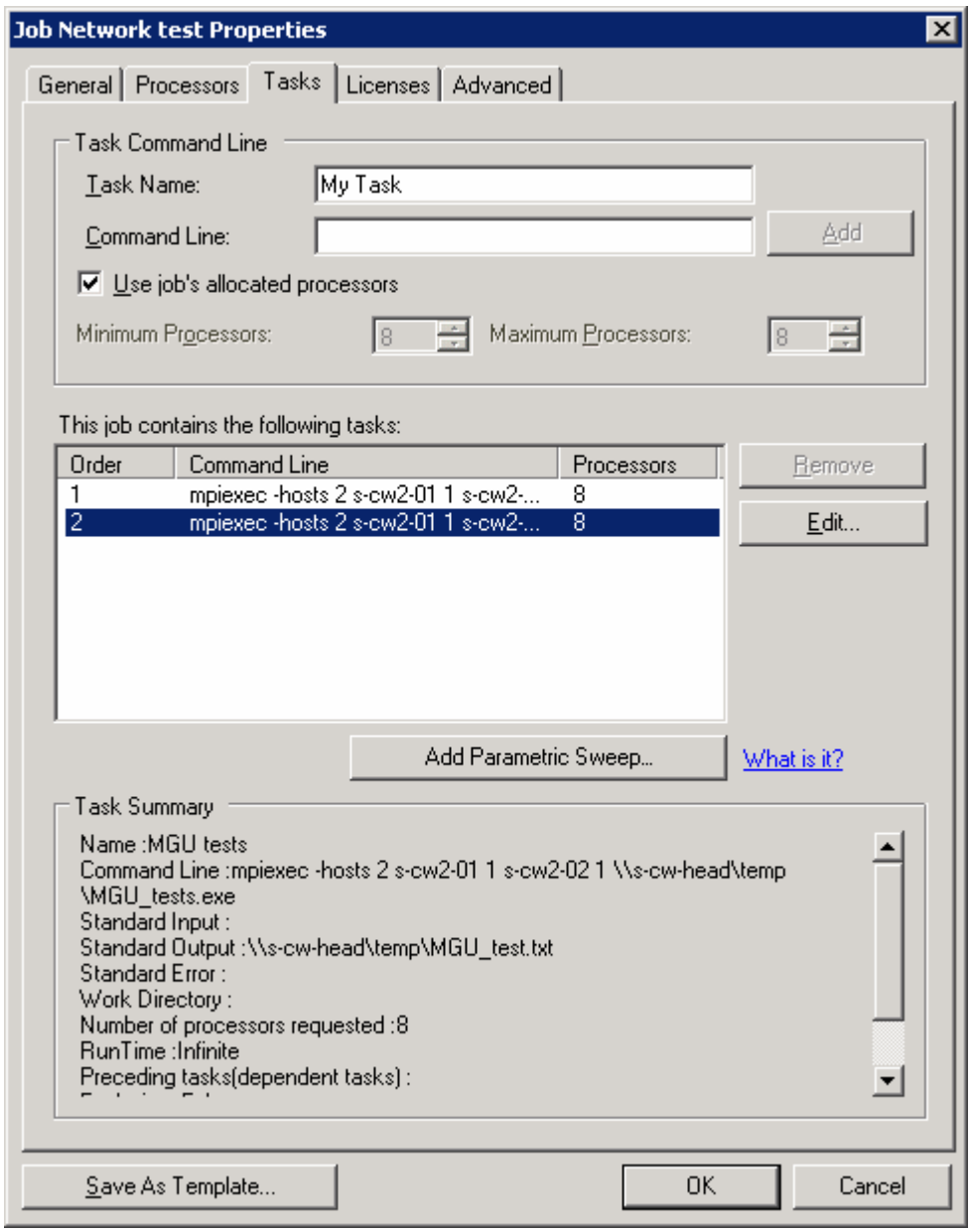
- Open **Job Manager**. Execute the menu option **File->Submit Job**… Give the task name and go to the window tab **Processors**,



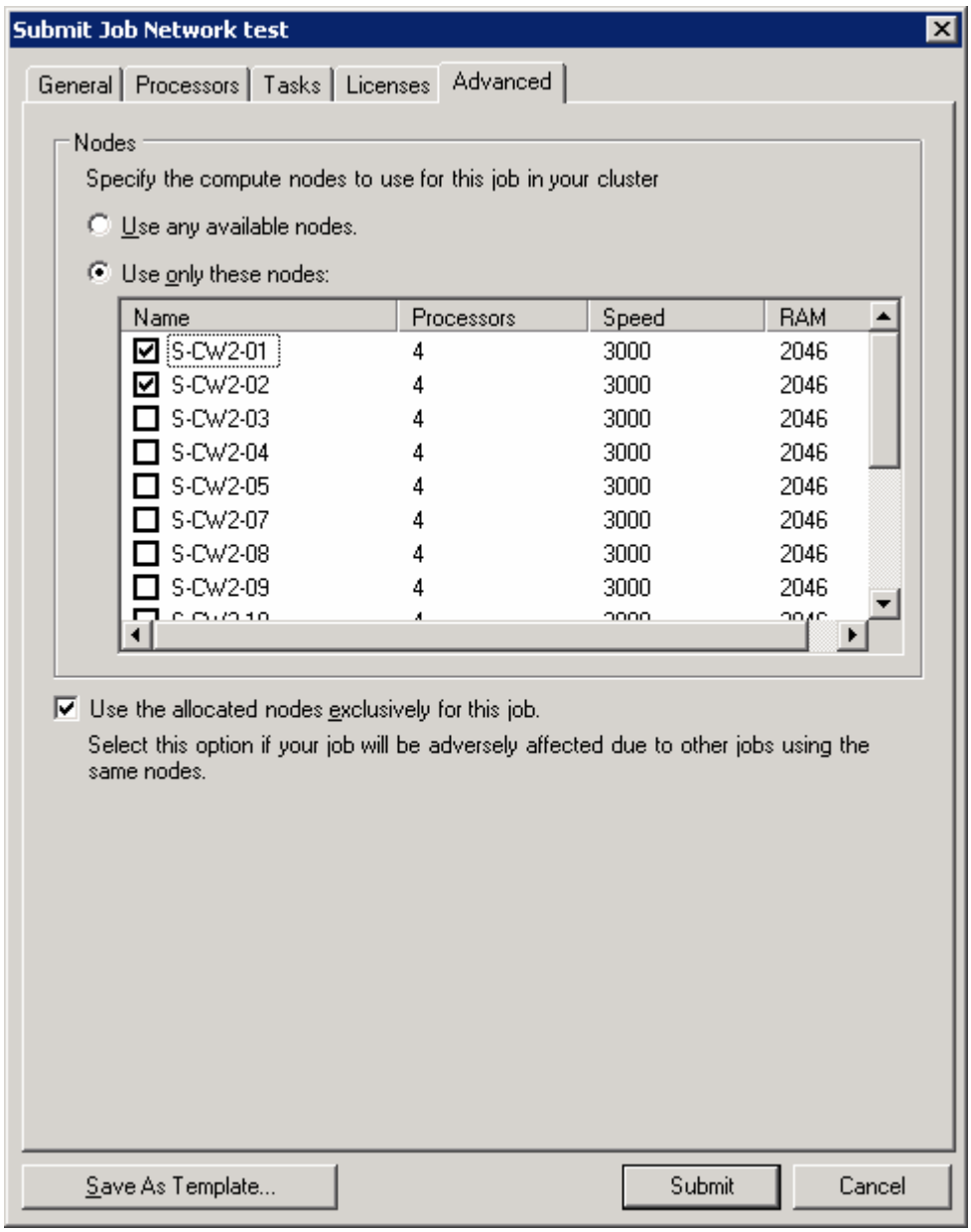
- Give the total number of processors on the 2 computer nodes, where you are going to execute the benchmarks (for instance, 8 in case you use 2 nodes consisting of 4 processors) as the job requirements. Go to the window tab **Tasks**,



- Add the two tasks to the job: "**mpiexec -hosts 2 s-cw2-01 1 s-cw2-02 1 \\s-cw-head\temp\imb.exe**", "**mpiexec -hosts 2 s-cw2-01 1 s-cw2-02 1 \\s-cw-head\temp\MGU_tests.exe**" (remember to change the command parameters for those corresponding to your case). The parameter "**hosts**" has the following format: "**n node1 m1 node2 m2 … noden mn**". You cannot use the environment variable **CCP_NODES** in this case, as only 1 process must be run on each node. Specify the files for redirecting the standard output stream for the tasks. Go to the window tab **Advanced**,



- Choose the option **Use only these nodes** and tick the nodes, which were given in the command at the previous step. Press the button **Submit** to add the jobs to the queue and enter the name and the password of the user authorized to run jobs on the cluster,



- The files, where the output stream was redirected to, contain the results of the benchmark execution. It is important to note that IMB carries out a number of various tests, but we are interested only in the first of them, **PingPong**, in order to obtain the network performance parameters. **PingPong** transmits data between two network nodes using the blocking functions *MPI_Send* and *MPI_Recv*, which is optimum for the estimation of the network parameters.

IMB_test.txt - Notepad

File Edit Format View Help

```
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000       131.62         0.00
            1         1000       130.68         0.01
            2         1000       130.47         0.01
            4         1000       128.91         0.03
            8         1000       127.64         0.06
           16         1000       124.63         0.12
           32         1000       124.85         0.24
           64         1000       123.77         0.49
          128         1000       123.78         0.99
          256         1000       125.24         1.95
          512         1000       128.76         3.79
         1024         1000       129.25         7.56
         2048         1000       137.03        14.25
         4096         1000       254.61        15.34
         8192         1000       364.53        21.43
        16384         1000       365.65        42.73
        32768         1000       491.61        63.57
        65536          640       880.04        71.02
       131072          320      1993.63        62.70
```

MGU_test.txt - Notepad

File Edit Format View Help

```
Size(b) Transfer (MB/sec)


Iteration 0
[0 -- 1] Latency: 127.272 microseconds (at 20 times)

1024    6.878
2048    8.065
3072    11.79
4096    15.71
5120    20.11
6144    22.67
7168    25.65
8192    26.35
9216    24.59
10240   26.89
11264   29.62
12288   32.4
13312   34.92
14336   37.72
15360   40.29
16384   42.98
17408   44.91
```

Discussions

- Define the terms “job” and “task”. What is the difference between them?
- What basic Microsoft Visual Studio 2005 settings should be specified for compiling a parallel program to be used in the environment MS MPI?
- What are the peculiarities of launching parallel tasks (compiled for MS MPI) on the cluster?
- What is the parametric sweep? What is the workflow?
- What parameters characterizing the network performance do you know? Define them.