

Implementation of Algebraic Equation in Distributive and Parallel Processing

Author Altaf Khan

Outline of technical report

Sr.No	Outline	Page No
1	Introduction.	2
2	Task Formation.	3
3	Description of Algorithm for Solving Task.	4
4	MPI Tool Installation	6
5	Description of Facilities Used for Implementation of The Task, Installation and Usage.	16
5.1	Use of MPI in Editor	17
6	Description of Developed Program (parts of the program, ways of interaction, synchronization, etc.):	17
7	Description of Conducted Test and Results.	24
8	Conclusion.	27
9	References.	27

1- Introduction:

In simplest sense parallel computing is a simultaneous use of multiple resources of system to solve complex problem. As compare to sequential computers parallel computers have multiple processor elements and problems are broken into sub parts and can be executed concurrently. Each part is further broken down into series of instruction. And instruction from each part execute simultaneously on different processors. So parallel computer is high performance computers and now days the demand of high computer is increased in different areas like: Atmosphere, Earth and Environment, nuclear, condensed matter, biotechnology and Genetics etc.

Main reason of using the parallel computing is to save time and money. In this theory more task are performed in short time using more resources, saving the potential cost. Second major purpose of developing parallel computer is solving complex and large problems. Many problems are so large and complex that is impossible to solve one computer so parallel computers are used to solve this types of problems like Grand Challenge requiring to petaFLOPS and petaBytes of computer memory [1]. Another beauty of parallel computer is providing concurrency; hence two tasks can run concurrently.

One of the more use of classification technique is called Flynn's Classification in this technique two dimension are used one is data streams and other is instruction streams. And each dimension has single and multiple states. Classification of Flynn's is:

- Single instruction and Single Data stream (SISD)
- Single instruction and Multiple Data streams (SIMD)
- Multiple Instructions and Single Data stream (MISD)
- Multiple instruction and Multiple Data streams (MIMD)

We will not discuss above terminologies in detail, mostly SIMD and MIMD terminologies are used for parallel processing. Now we are going to focus on our project.

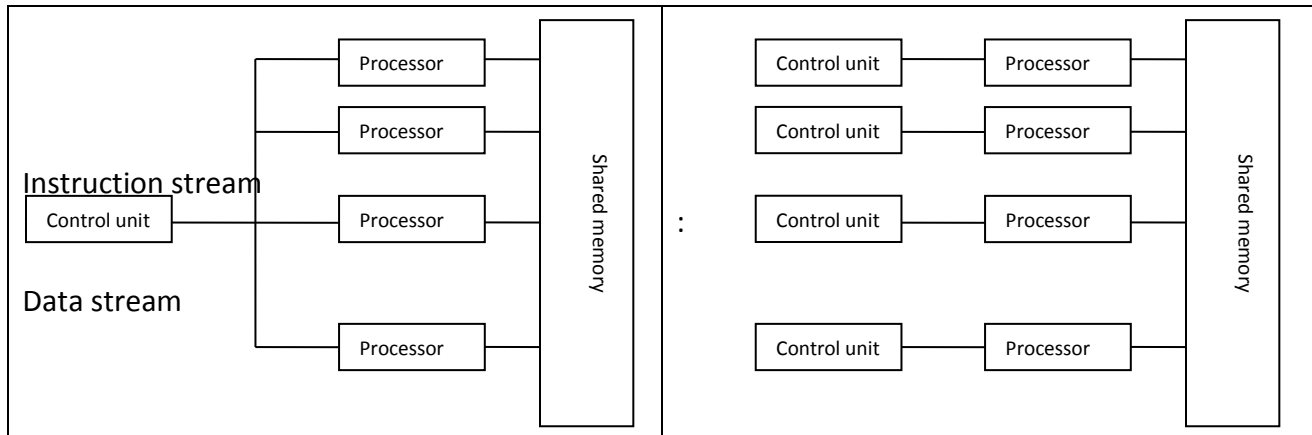


Figure 1.1 SIMD and MIMD Architecture.

Now we shall be illustrated to project. Definition of our project is implementation of algebraic equation on two different physical processors which should be worked concurrently. First divide my project to different tasks and then we solve these tasks, discuss below.

2- Task Formulation:

Parallel processing is most common among number of processors. Our current computing technology work on it to fulfill the requirement of market. No one have much time, so parallel processing is way in which we can reduce the time of execution. This project is to compare the result of sequential and parallel processing time, and then we can find benefit of parallel processing. Actually purpose of this task is to show the speedup and efficiency of processing and we can easily then observe that parallel processing is much faster as compare to sequential processing. But due to this we have to execute complex task otherwise we can't differentiate between sequential and parallel execution. So we use multiplication and summation product to solve this problem and we got the results which are mentioned below.

My problem statement is summation and multiplication result. And after getting result multiply/ add/ divide etc both values. My task formula is shown here. First part is use for addition of ith value. For example $i=0$ and $N=10$, so $\text{result1} = 0+1+2+3+4+5+6+7+8+9+10$. First 10 number will sum and other is multiplication $M=10$ so $i=1$ start point and 10 is end point. $\text{Result2}=1*2*3*4*5*6*7*8*9*10$. Now bother result is $\text{result1} * \text{result2}$ if opt value =*.

So for I create two array one for addition and one for multiplication. Array_sum[i] and multi_array[i]. During execution it will work just like a **sum= sum +Array_sum[i]** if ($0 \leq i \leq n$). In addition multiplication is perform same **mul= mul*multi_array[i]** if ($0 \leq i \leq n$). At the end

Result is **result= (sum) Opt(mul)** and opt = *,+,-,^,/ .

$$\left(\sum_{i=0}^N A_i \right) \text{opt} \left(\prod_{i=1}^M B_i \right)$$

In this equation we divide xi rows to parallel segments of junk which is equal to numbers of processors. And each processor solves division parts of program. Sending part contains starting value and ending value of partial array which will executed on child process. After completing execution child process will return back part of array result to his master or root processor. We will describe more in algorithm step.

3- Description of Algorithm for Solving the Task:

The algorithm is as follow for this equation(using C++ MPI):

$$\sum_{i=0}^N (A_i) \text{Opt} \prod_{j=1}^M (B_j) \quad \{ N=1,2,3,4\ldots n, M=1,2,3\ldots n, \text{opt}=*,+,- \text{ etc} \}$$

Sum= sum+Ai and Multiply is Mul= Mul*Bj.

Get number of processors by gets size command and ranks of processors

1- Condition: if processor rank =root processor

- Get value "Num_rows_sum" of N for summation of N values
- Get value "Num_multiple " of M for multiplication of M values
- Get Opt for perform operation on both results.
- If both values > **1000000** than show value is so long.

- Else go away
- Initialize Array 1 and array 2 .(Array[i] = i+1 and multiple = i+1);
- Get average for each processor if **an_id < processors**
 $\text{avg} = \text{Num_rows_sum} / \text{processors}$ and $\text{avg2} = \text{num_Multiple} / \text{processors}$

Parallel process sending data to each child:

- Note start time of parallel processing (**starttime**)
- **An_id < processors**
- Get start **row = an_id * avg_row_sum + 1**
- **Endrow = an_id * avg_rows_pr_process**
- If **num_row - end_row < avg_rows** than **endrow = num_row - 1**
- Send number of rows to processor: **num_row_send = endrow - startrow + 1.**
- Send to processor which have id = **an_id**. (using **MPI_Send**)
- Send array value with "start value " tage. (using **MPI_Send**)
- Increment **an_id**.
- Repeat up till **processor < process_num**.
- Sending data for multiplication:
- Get average rows of array
- Start value
- End value (using above procedure)
- Send to **an_id**. (using **MPI_Send ()**).
- Increment **an_id** and **an_id < process_num**.
- Repeat above end to all processors.

Else

Receiving data from master (child will get data):

- Receive number of rows from master / root processors. (**MPI_Receive();**)
- Receive data (rang of array)
- Manipulate it with operator (* or +) and I <**receive_rows**>.

- Increment of `l` with 1 and so on..
- Complete calculation until `< receive_rows`
- return to master with MPI Send operation.
- Close process at child.

Receiving Data at Master / root processor:

- Receive data from child with **MPI_Recv** () function from every child,
- Collect all child data for summation(partial sum).
- Add summation result **sum= partial sum + root sum**
- Collect all child data for multiplication (partial multiplication).
- Multiply root result with child (**mult= root_Mult*partial_mul**).
- Final **result** = (multiplication) operator(summation)
- End time (**wtime()**) of parallel process.
- Difference of time **TP= endtime- starttime**.
- Show **result** and **Time** (seconds)
- Close MPI. (Finalized all operations).

4- MPI Tool Installation :

First of all we installed software which is used for parallel and distribute programming. There are many software available in market but we used the best for our project and operating system. We are using window 7 so first we got open source MPI CH2 and Microsoft HPC PACK 2012 both tools are working on window platform so any one can use these tools for parallel processing. We worked on MPICH2. MPICH2 is open source so you can easily download from internet. <http://www.mcs.anl.gov/research/projects/mpich2staging/goodell/downloads/index.php?s=downloads>. This link of MPICH2, in this page you will find lot of version available for different platforms but we downloaded it for window operating system. So installation step of above mpich2 is described below:

1. We connected two PCs to each other, install MPICH2 on each but when you are going to install be careful that your system firewall should be off and each pc should be part of network or point to point pc connection.
2. Then installed this above download mpi program simply using window user interface but system should be under the administrator rights. So I will use command prompt to install mpich2.
3. During installation select MPICH2 to be installed for everyone.
4. Use default setting sitting during installation. Steps of installation below:

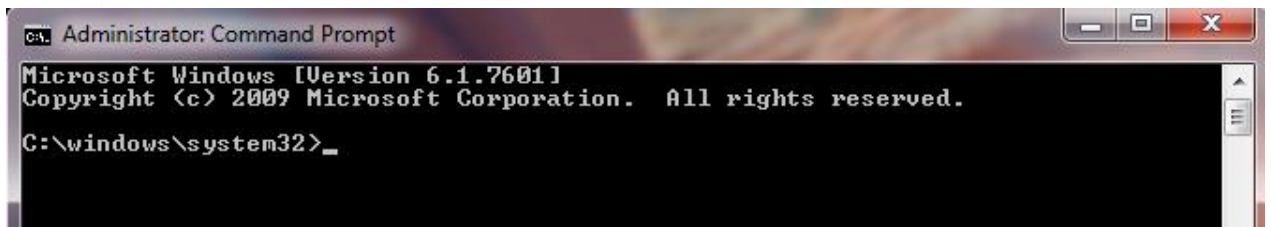


Figure 1.2 Administrator command prompt.

5. Next to execute program

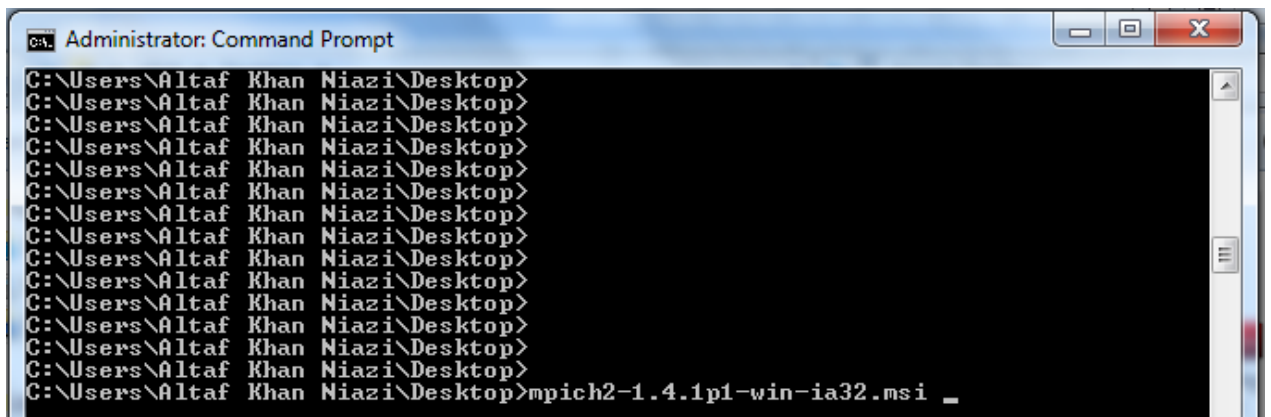


Figure 1.3 mpich2-1.4.1p1 is ready to install

6. Installation is starting.

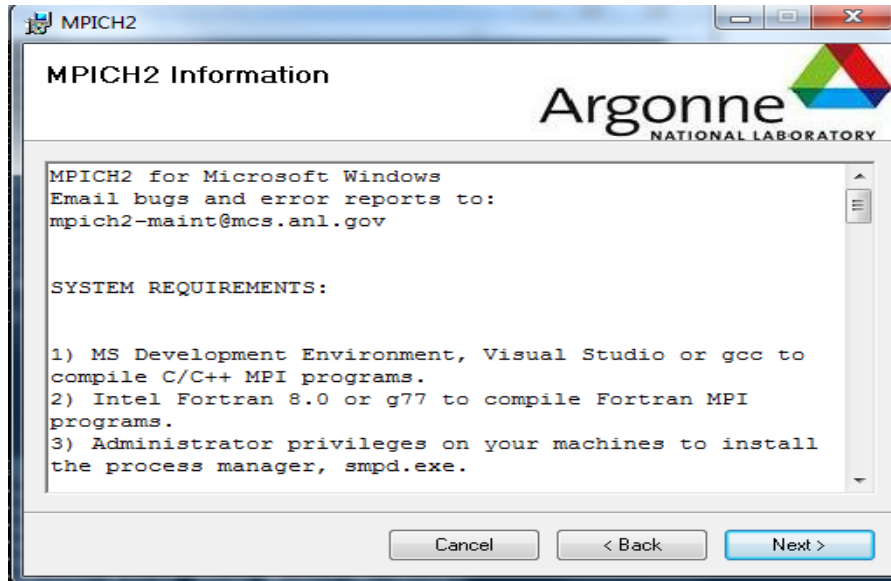


Figure 1.4 mpich2 is installing

7. Next to complete wizard

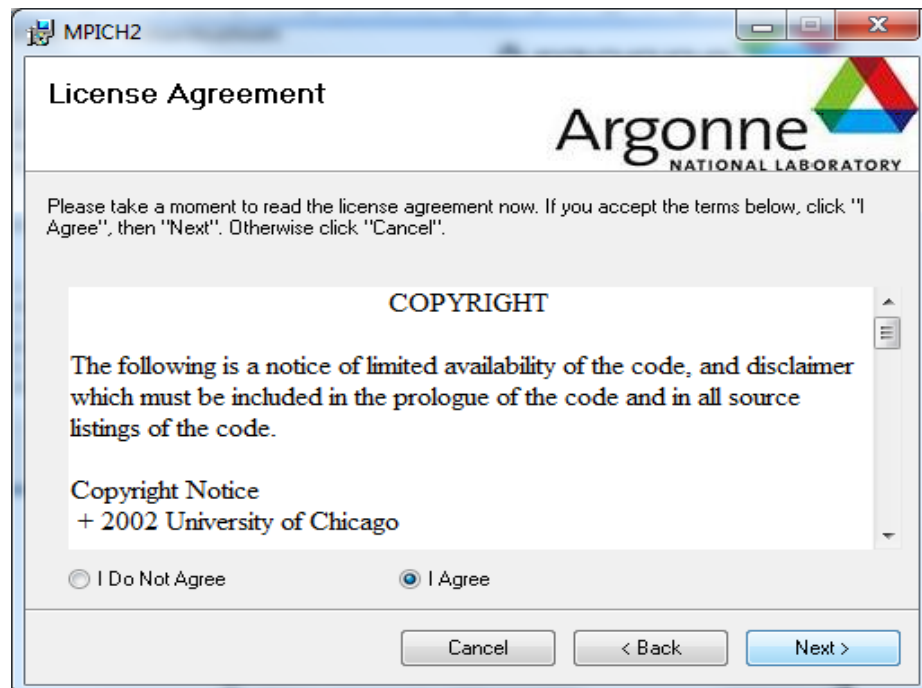


Figure 1.5 I agree and do not agree copyright

8. Process manager setup.

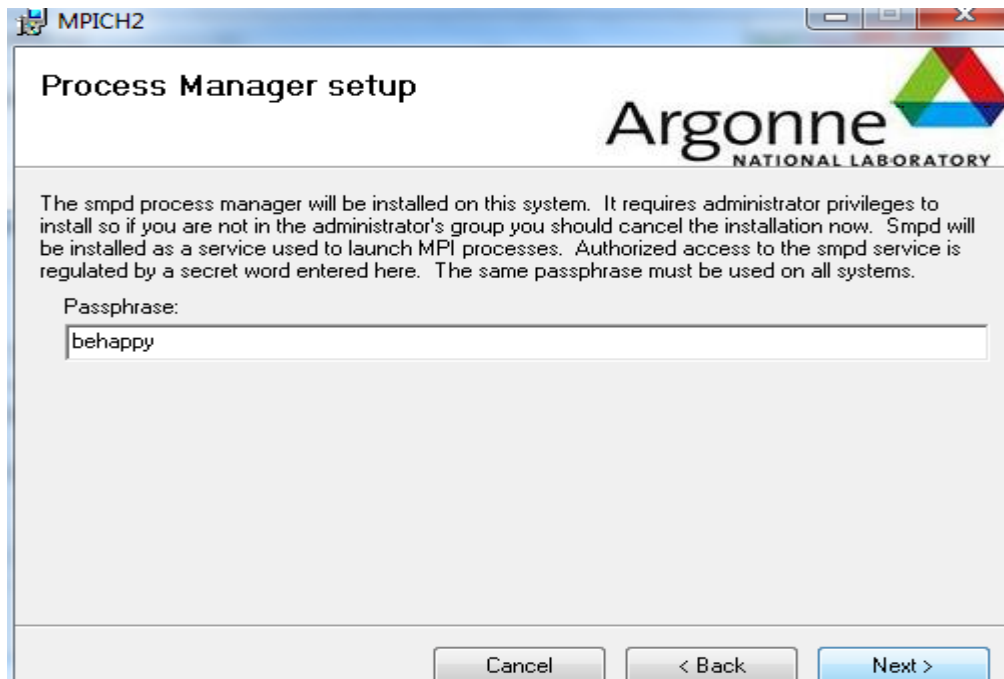


Figure 1.6 process manager setup

9. Select installation folder. And NEXT

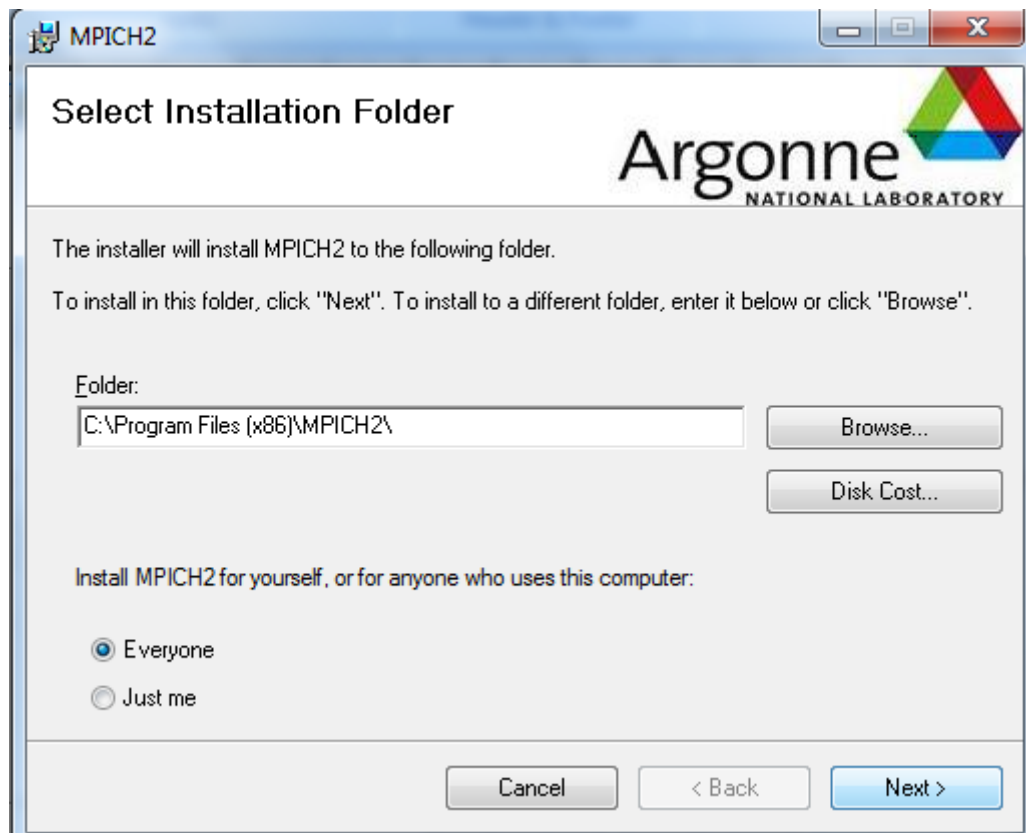


Figure select installation folder for everyone.

10. Mpich2 installing to end

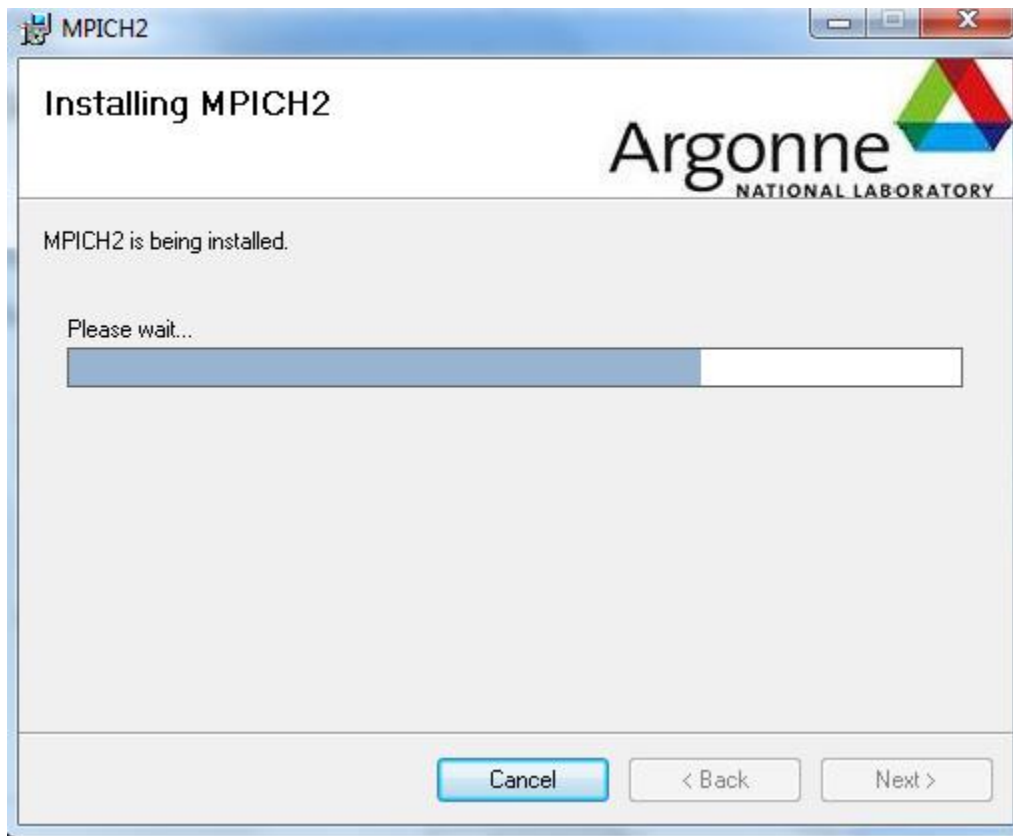


Figure 1.6 during installation

11. Installation has completed close the wizard.

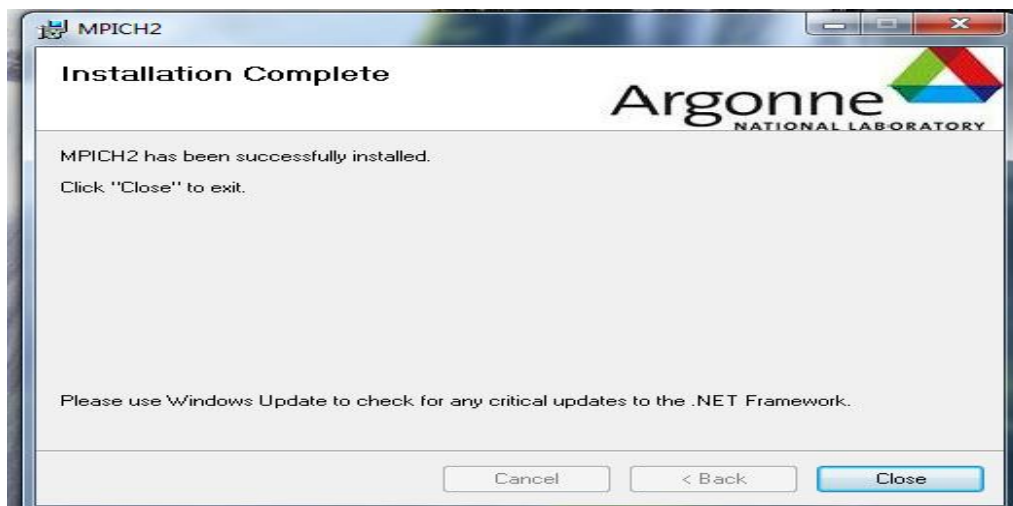
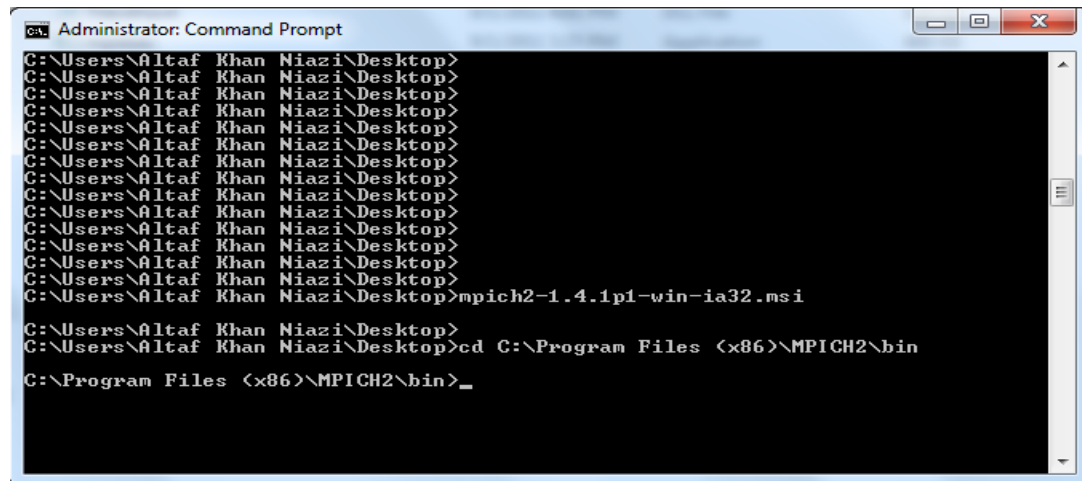


Figure 1.7 Installation is completed of MPICH2.

12. After completing installation close, open command prompt window with administrator.

13. Go to specific folder. We are using 32bit so the path is C:\Program Files(x86)\MPICH2\bin.



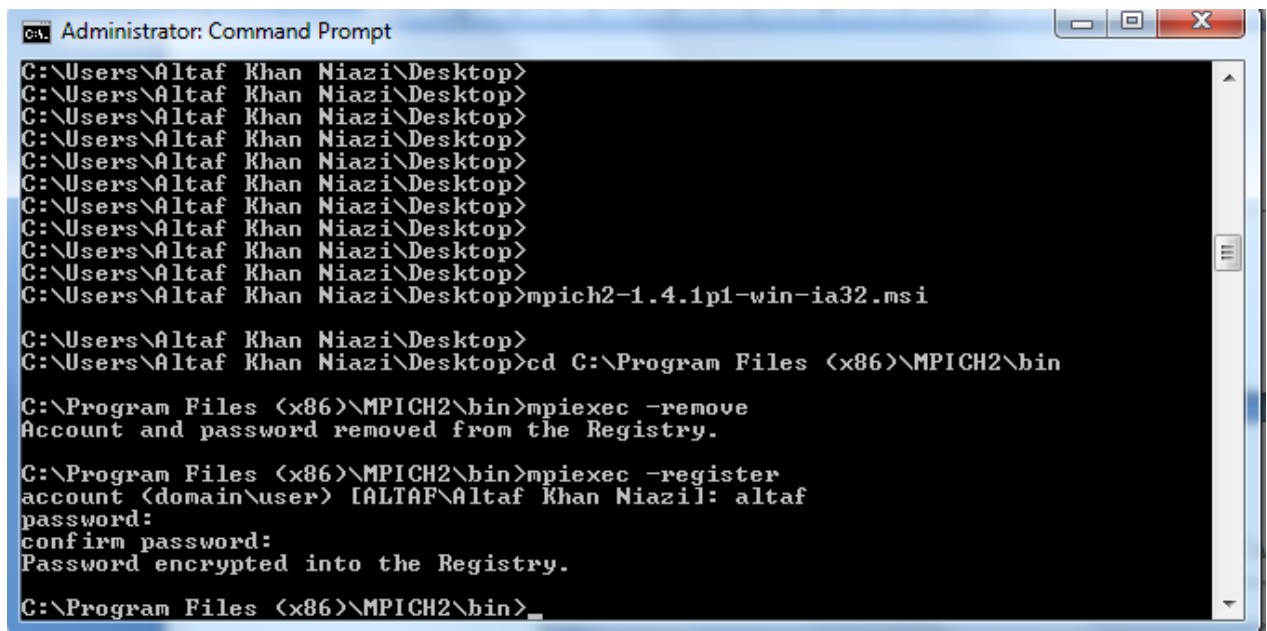
```

Administrator: Command Prompt
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>mpich2-1.4.1p1-win-ia32.msi
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>cd C:\Program Files (x86)\MPICH2\bin
C:\Program Files (x86)\MPICH2\bin>_

```

Figure 1.8 get the mpich2 path in CMD

14. Mpiexec -remove
15. Mpiexec -register



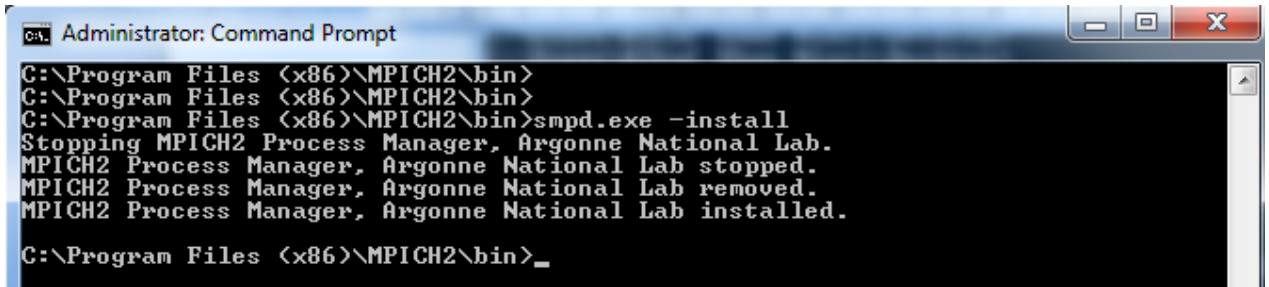
```

Administrator: Command Prompt
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>mpich2-1.4.1p1-win-ia32.msi
C:\Users\Altaf Khan Niazi\Desktop>
C:\Users\Altaf Khan Niazi\Desktop>cd C:\Program Files (x86)\MPICH2\bin
C:\Program Files (x86)\MPICH2\bin>mpiexec -remove
Account and password removed from the Registry.
C:\Program Files (x86)\MPICH2\bin>mpiexec -register
account <domain\user> [ALTAf\Altaf Khan Niazi]: altaf
password:
confirm password:
Password encrypted into the Registry.
C:\Program Files (x86)\MPICH2\bin>_

```

Figure 1.9 registration of mpiexec.

16. Enter `smpd.exe -install` in command prompt



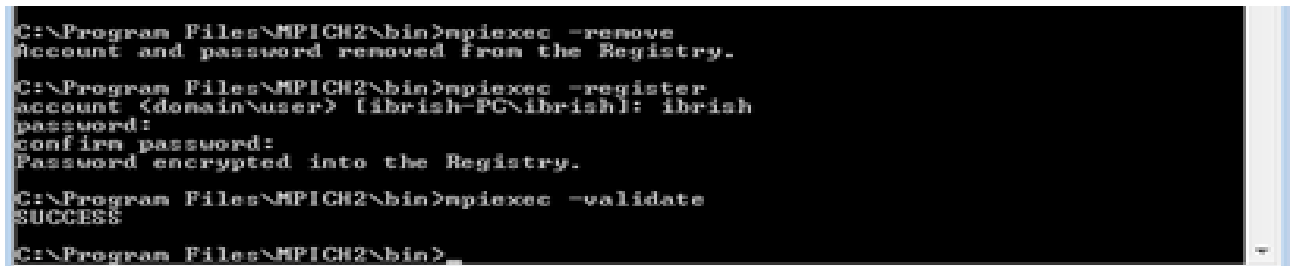
```

C:\Program Files (x86)\MPICH2\bin>
C:\Program Files (x86)\MPICH2\bin>
C:\Program Files (x86)\MPICH2\bin>smpd.exe -install
Stopping MPICH2 Process Manager, Argonne National Lab.
MPICH2 Process Manager, Argonne National Lab stopped.
MPICH2 Process Manager, Argonne National Lab removed.
MPICH2 Process Manager, Argonne National Lab installed.

C:\Program Files (x86)\MPICH2\bin>_

```

Figure1.10 check the validation.



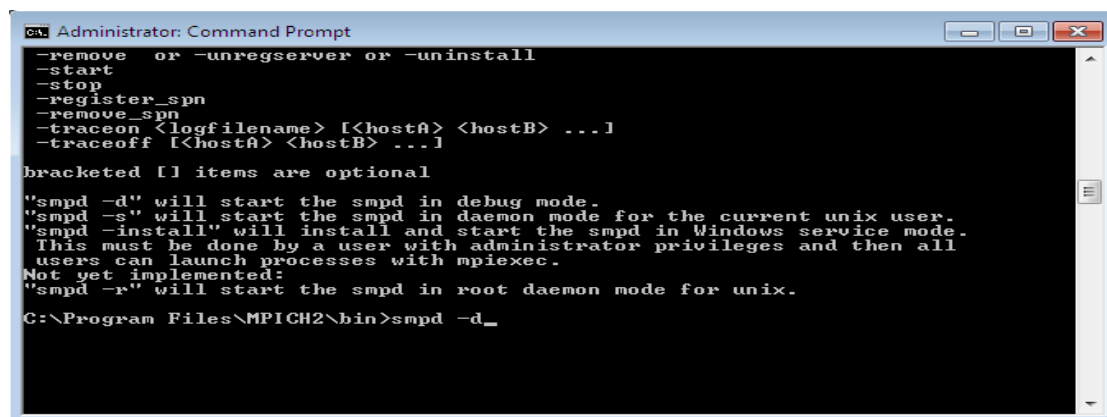
```

C:\Program Files\MPICH2\bin>mpicxxec -remove
Account and password removed from the Registry.
C:\Program Files\MPICH2\bin>mpicxxec -register
account (domain\user) [ibrish-PC\ibrish]: ibrish
password:
confirm password:
Password encrypted into the Registry.
C:\Program Files\MPICH2\bin>mpicxxec -validate
SUCCESS
C:\Program Files\MPICH2\bin>_

```

Figure 1.11 validation status is success .

17. Help and other option of `smpd.exe`



```

Administrator: Command Prompt
- remove or -unregserver or -uninstall
- start
- stop
- register_spn
- remove_spn
- traceon <logfile name> [<hostA> <hostB> ...]
- traceoff [<hostA> <hostB> ...]

bracketed [] items are optional

"smpd -d" will start the smpd in debug mode.
"smpd -s" will start the smpd in daemon mode for the current unix user.
"smpd -install" will install and start the smpd in Windows service mode.
This must be done by a user with administrator privileges and then all
users can launch processes with mpicxxec.
Not yet implemented:
"smpd -r" will start the smpd in root daemon mode for unix.

C:\Program Files\MPICH2\bin>smpd -d_

```

Figure 1.12 `smpd -help` and `-d` function

```

C:\Administrator: Command Prompt
[00:24841]..\smpd_set_smpd_data
[00:24841]..\smpd_set_smpd_data
[00:24841]..\SMPDU_Sock_create_set
[00:24841]..\SMPDU_Sock_get_sock_set_id
[00:24841]..\SMPDU_Sock_get_sock_set_id
[00:24841]..created a set for the listener: 492
[00:24841]..\SMPDU_Sock_listen
[00:24841]..ERROR:Error binding socket to given port, Only one usage of each so
cket address <protocol/network address/port> is normally permitted. <10048>
[00:24841]..ERROR:Error creating listen sock
[00:24841]..\SMPDU_Sock_listen
[00:24841]..ERROR:SMPDU_Sock_listen failed,
sock error: Error = 10048
[00:24841]..\smpd_entry_point
[00:24841]..calling SMPDU_Sock_finalize
[00:24841]..\SMPDU_Sock_finalize
[00:24841]..\SMPDU_Sock_finalize
[00:24841]..\smpd_exit
[00:24841]..\smpd_kill_all_processes
[00:24841]..\smpd_kill_all_processes
[00:24841]..\smpd_finalize_drive_maps
[00:24841]..\smpd_finalize_drive_maps
C:\Program Files\MPICH2\bin>

```

Figure 1.13 smpd -d function which you can use

18. Ip config to get IP address of computer

```

C:\Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Windows\system32>ipconfig

```

Figure 1.14 ipconfig for checking the ip address

19. Now click start and open wmpiconfigure file .

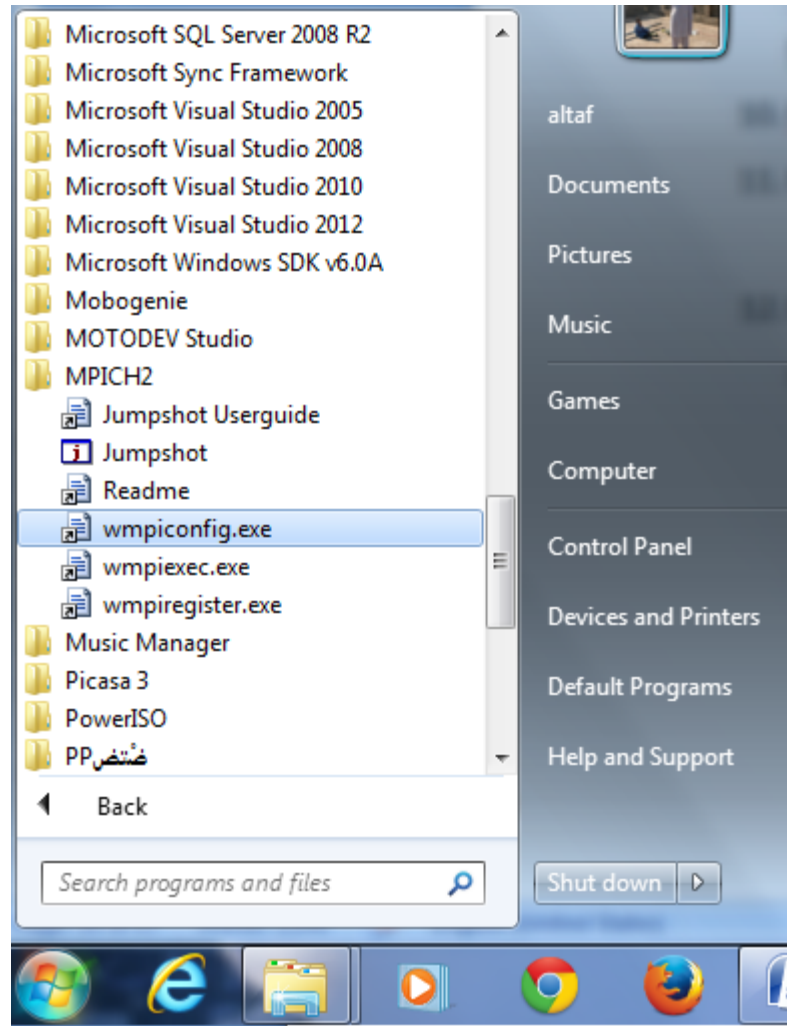


Figure 1.15 Wmpiconfig file and wmpixec file can open here.

20. Configuration of wizard

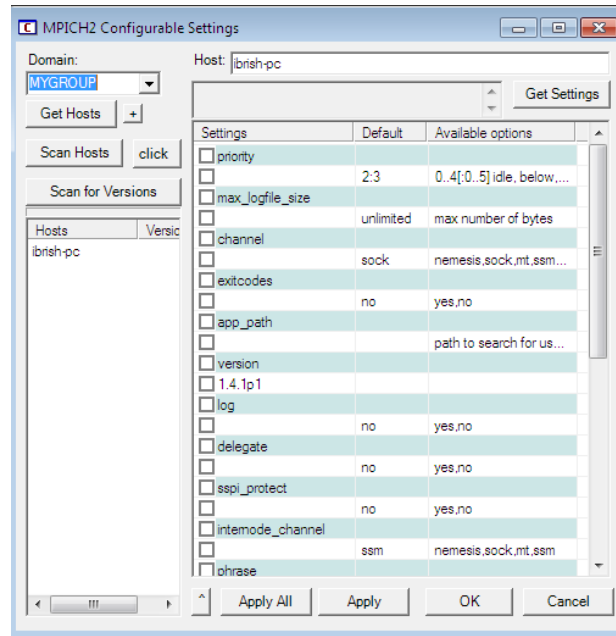


Figure 1.16 configuration wizard

21. Get host by domain and scan host, setting the host address.

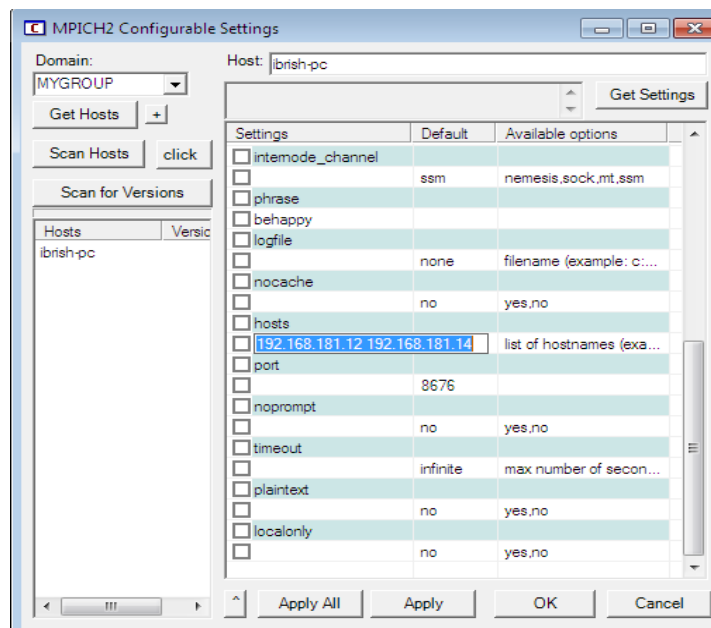


Figure 1.17 configuration wizard with set ip address and get host list and Apply All

22. Now open click start button and select wmpiexec and run this then you will get execution environment for parallel process.

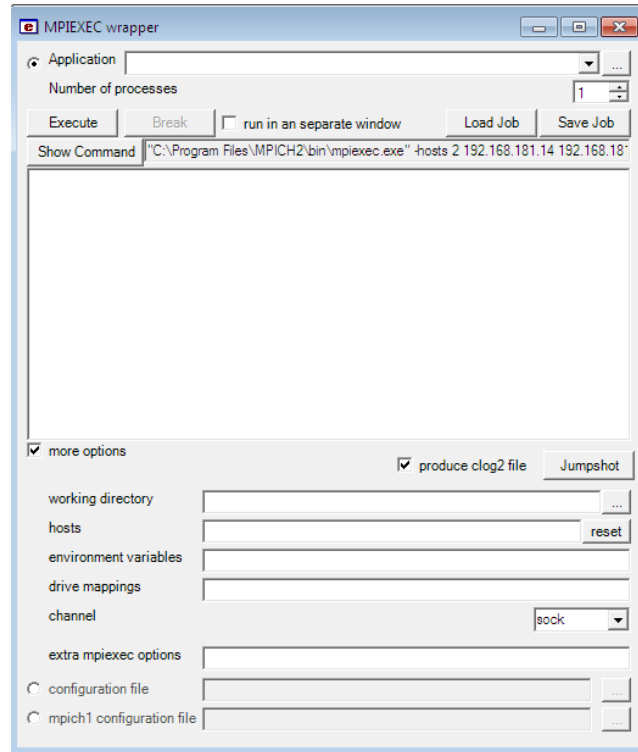


Figure 1.18 mpiexec environments,

In this environment, application button for application path which you want to execute. Number of processor is processors which are participate in execution. Hosts are address of processors. Channel is a drop down list which we can use during process. By default we have to use sock it is for parallel processing and shm for sequential execution task on one processor etc.

5. Description of Facilities Used for Implementation of the Task, Installation and Usage:

Implementation of parallel processing on two computers we used two type of equipment illustrate below:

1. Hardware:
 - a. Two PCs (Ram 1 GB at least of each pc)
 - b. Network cat 6 cable for communication.
2. Software:
 - a. Window 7 operating system

- b. Visual studio 2010 c++
- c. MPICH2 (also you can use DeinoMPI/Microsoft HPC pack 2012 tool).

5.1 Use of MPI in Editor:

There are different step to installation and usage (procedure is shown in above installation of MPI tool)

1. User can install it by "parallel.Exe" file and get icon in start menu of parallel processing.
2. If you get error then install using this path:C\.. \obj\x86\Debug. Double click on exe file and then it starts automatically.
3. After installation user run this program first **click Start** button-> **click parallel**-> run.
4. This is work for parallel so you have to use MPICH2 tools
5. Set path
6. Set number of processors.
7. Set channel
8. Click on Execute
9. Enter number of values (rang of summation and multiplication). See the results.
10. Calculate speed up and efficiency and check performance of parallel processing.

6. Description of Developed Program (parts of the program, ways of interaction, synchronization, etc.):

This program is designed to check the parallel processing on two PCs and get result that it is better work as compare to sequential PC. We can observe the performance of parallel processing on distributed manner. Main purpose of this work is to get speedup and efficiency of parallel processing compare to single sequential PC.

How to work?

First of all when you want to execute exe file you have to make sure that smpd is working on system (MPI CH2).

How to use MPI?

MPI is message passing interface which is develop for specially parallel machine. MPI is open source and it is easily available. MPI library is used for develop program. Above I mention MPI installation on window. So how to use in code I will explain now .

First of all visual studio should be install on your pc and then you can set the directory setting .I show in steps.

1. Open project in Visual studio click project =>and click on properties.

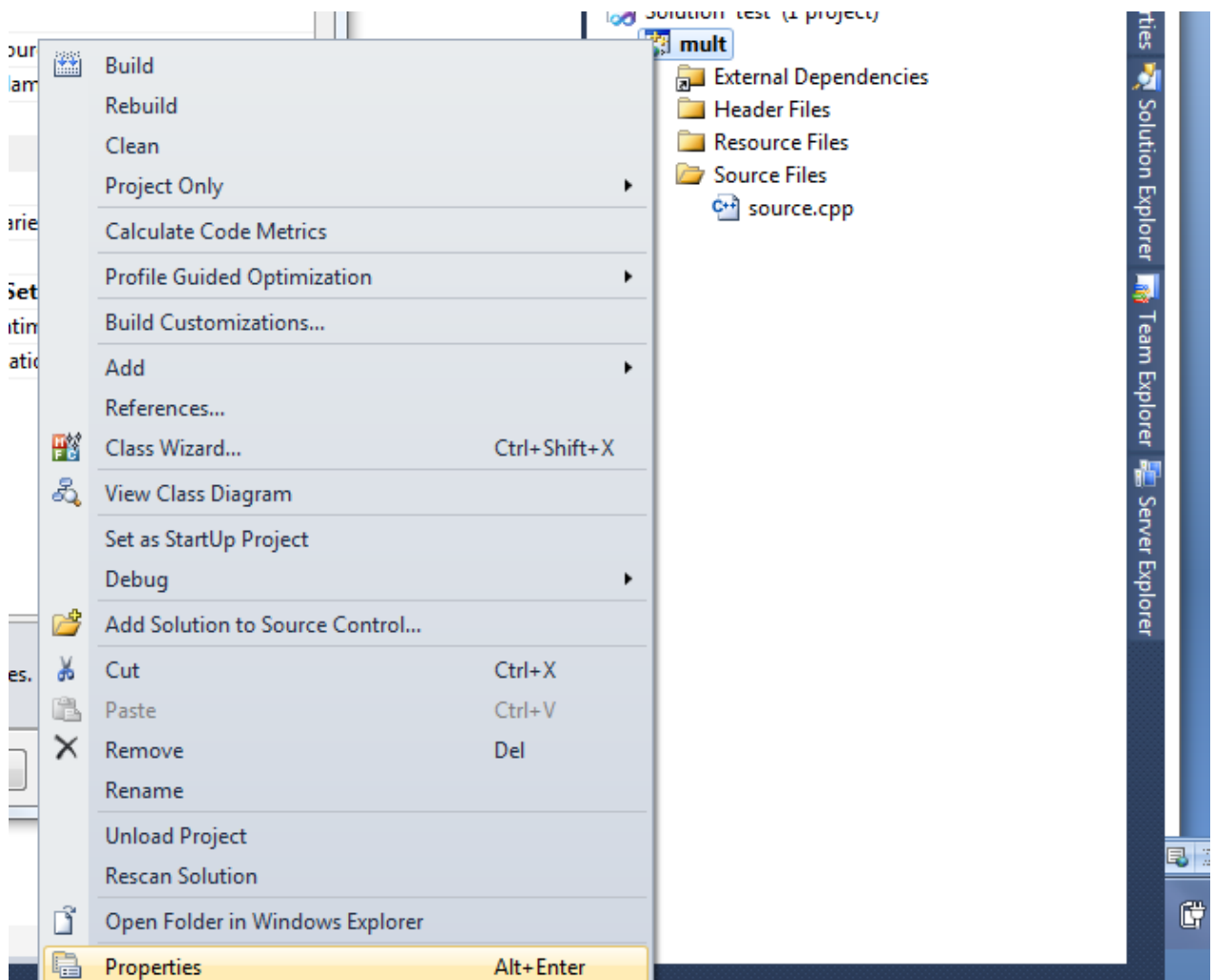


Figure 1.19 project properties to set the path of MPI Library files.

2. Open properties and click c/c++ option and set directory

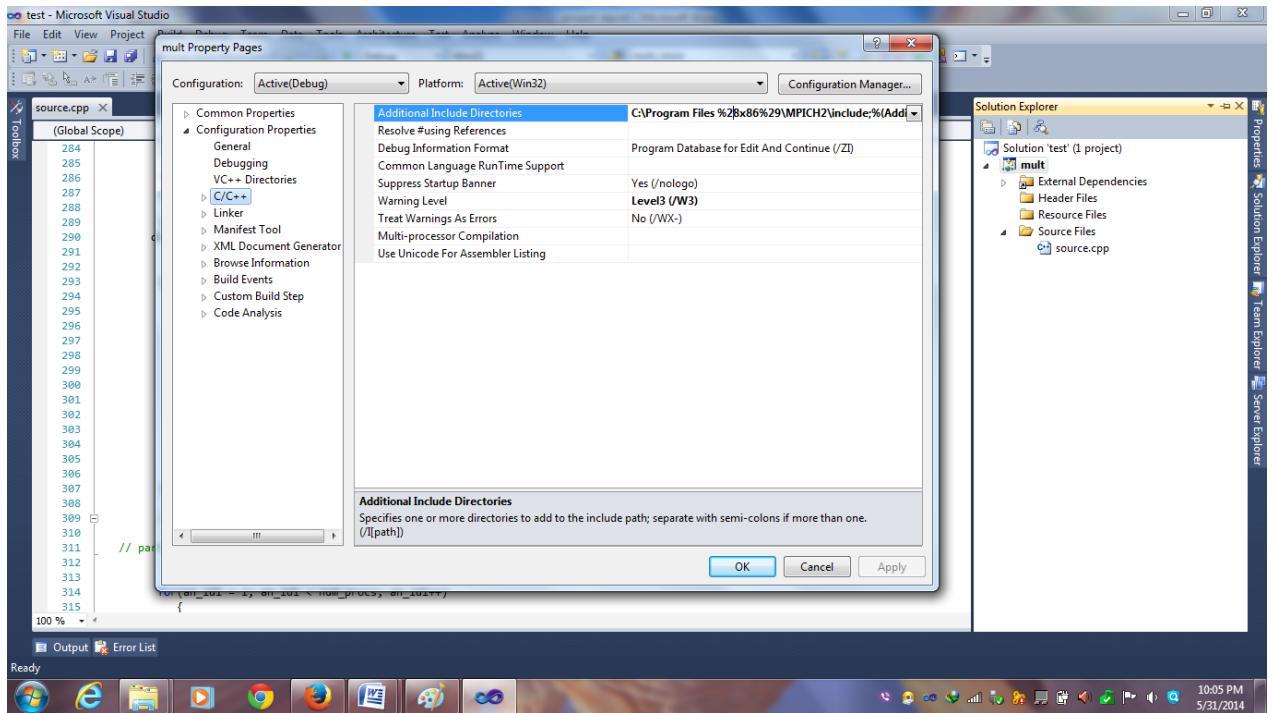


Figure 1.20 c/c++ addition dictionary directory.

3. Click addition directory and set path of directory.

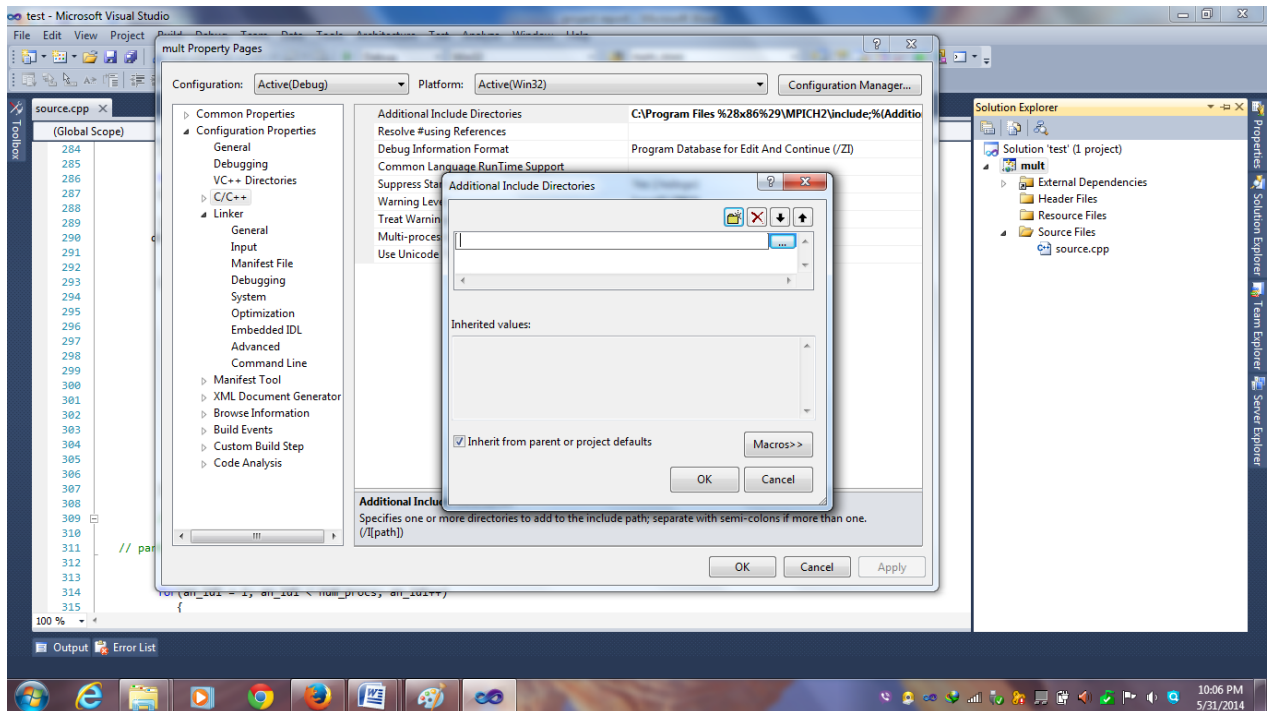


Figure 1.21 click edit and add include folder of MPI and click ok to save this link

4. After adding directory click linker and add addition library from bin of mpich2.

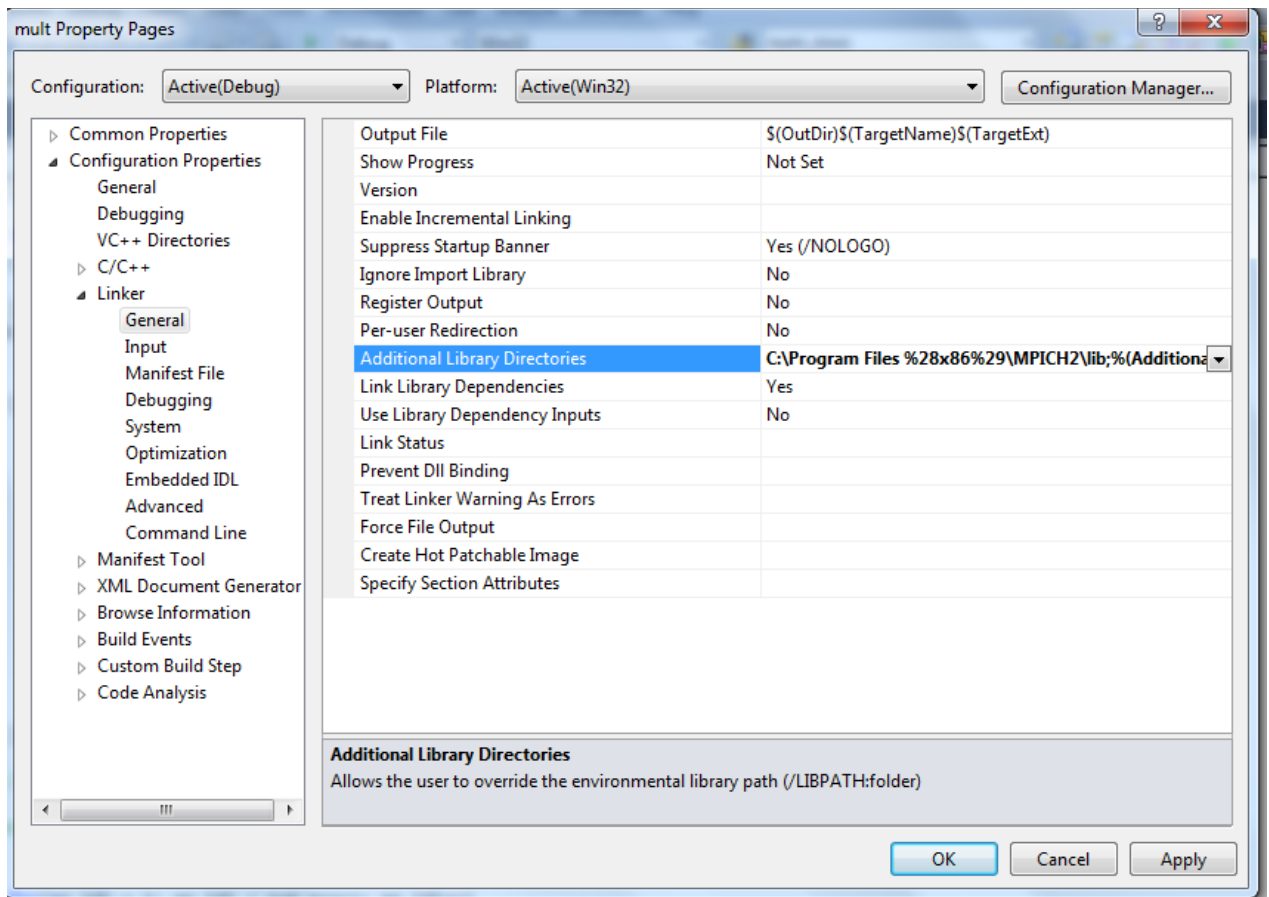


Figure 2.22 add library file in link from lib folder of mpich2.

5. Click linker -> input and set depend field (mpi.lib and cxx.lib) files.

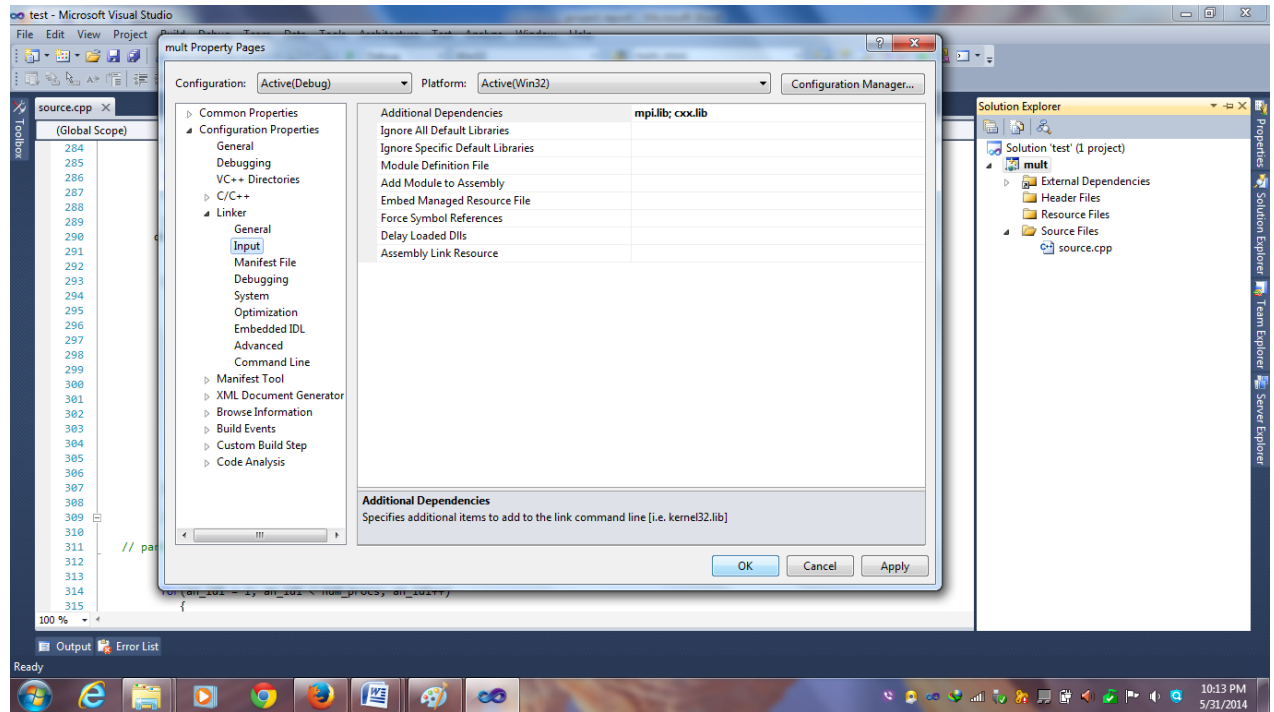


Figure 2.23 click input and add dependencies mpi.lib, cxx.lib file in top text box. Apply to all.

6. Apply to all and ok. Now machine is ready for programming.

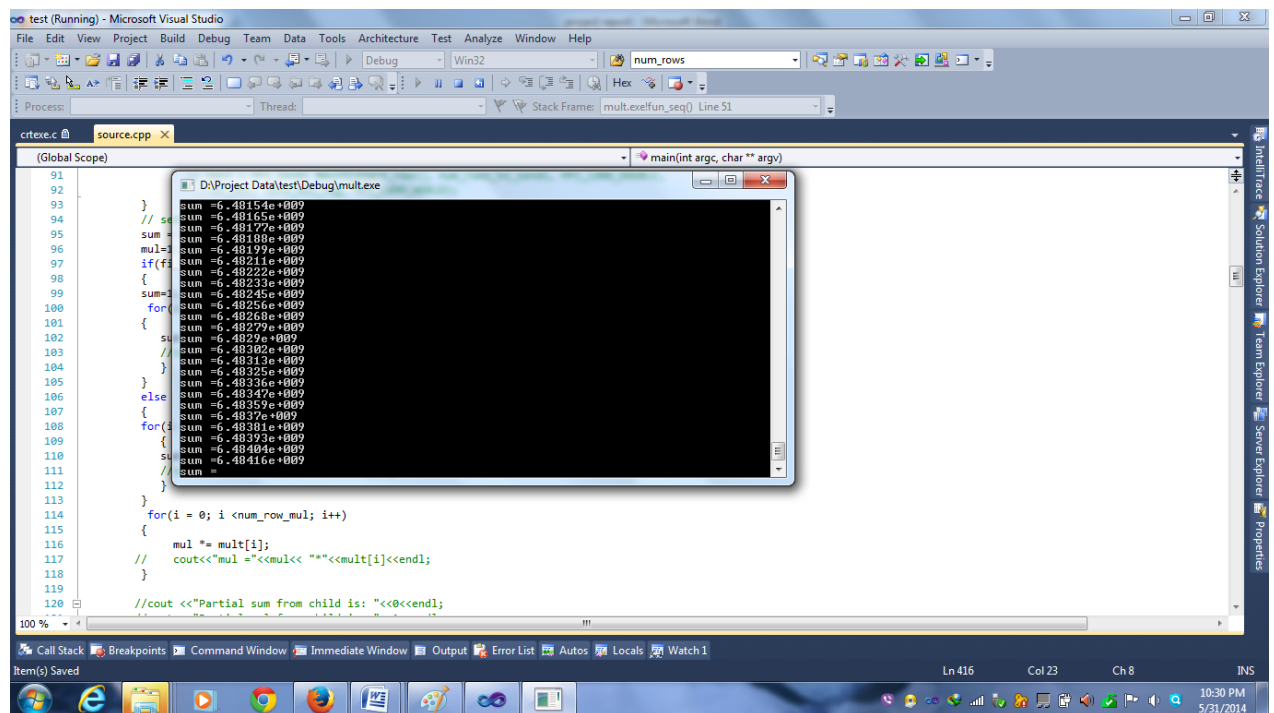


Figure 2.24 add code using MPI and debug this and make exe file for parallel processing.

7. Build program and execute this. Now go project folder where you save and open debug folder and get exe file for parallel processing.

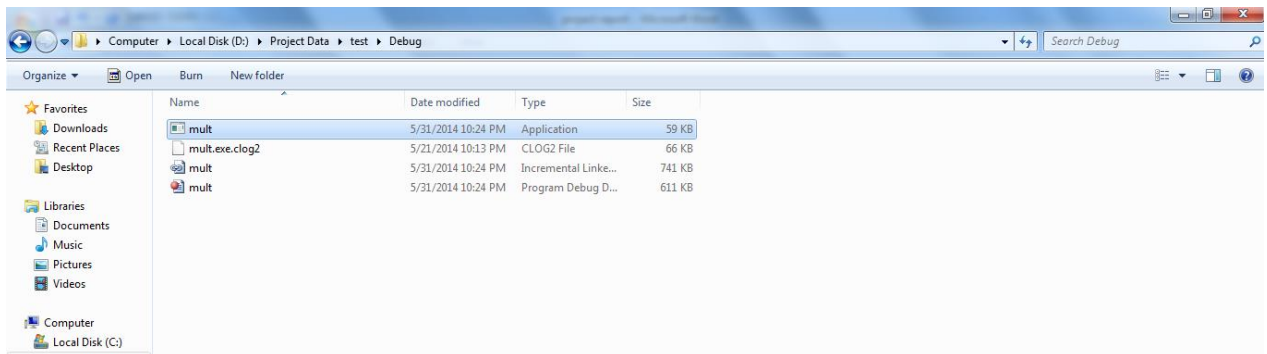


Figure 2.25 exe file in debug folder copy it and past in public share folder and execute it on mpich2 wmpiexe.

8. Now run exe file on above MPICH2 plate form. Result is shown here.

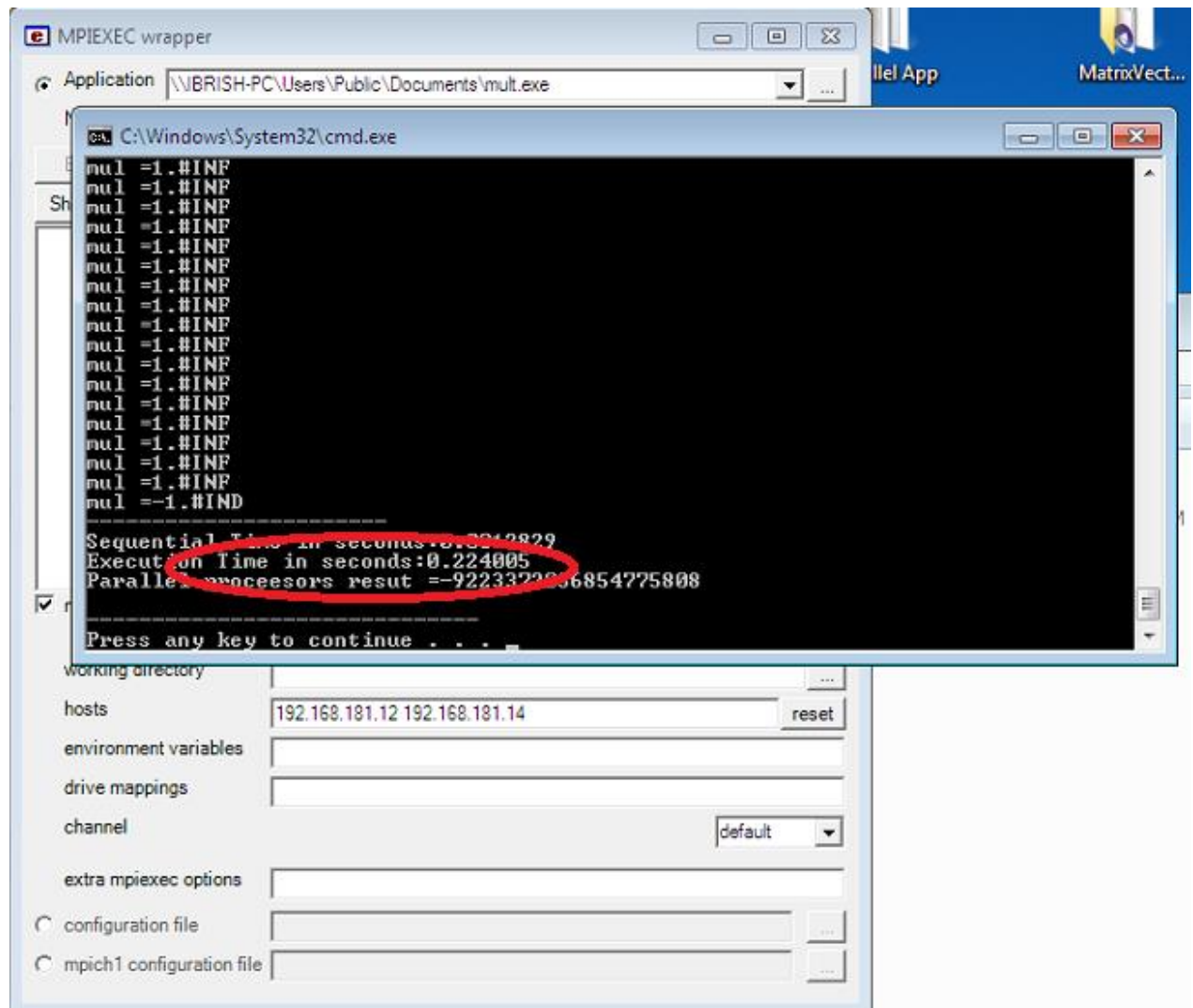


Figure 2.26 result of one processor in seconds. And output .

My result is #inf because I have entered 6000 input number long number so in multiplication it is very long number so it show – result but the execution is T1= 0.22400 for one processor.

9. When number of processors is = 2

First of all we found execution time of processor one and then add second and get two p1 and p2 values. After this we change value 2000 by 2000 and get again both values. Repeat same procedure for different 4 values.

13	Sr.No	Input N	Input M	p1(second)	p2 (second)	Speed up (tp1/tp2)
14	1	1000	1000	0.03825	0.0428	0.893691589
15	2	2000	2000	0.05729	0.04469	1.281942269
16	3	3000	3000	0.1296	0.07572	1.711568938
17	4	4000	4000	0.1504	0.079	1.903797468
18						

Figure 1.28 Execution time of p1 and p=2 processors and speed up

After finding execution value, we found speed up which is found by $sp = T_1/T_p$ $T_1 = p_1$ execution time and $T_p = p_2$ execution time. Now we calculate efficiency and fraction values.

	Efficiency p1/p2	fraction (T1 and p2)
	0.446845794	1.237908497
	0.640971134	0.560132658
	0.855784469	0.168518519
	0.951898734	0.050531915

Figure 1.29 Efficiency of executed result and fraction values

Now We found efficiency of Processors using $Ef = (Sp/Num_processors)$. and fraction is

calculated by $f = \frac{T_p - T_1 / p}{T_1(1 - 1/p)}$ this formula. When we increase number of N efficiency is increased and fraction is following also above same N values. We show this result in graph. First graph show execution time of two processors. Second one is for speed up and last for efficiency. So you can easily understand to observing the result of all graphs.

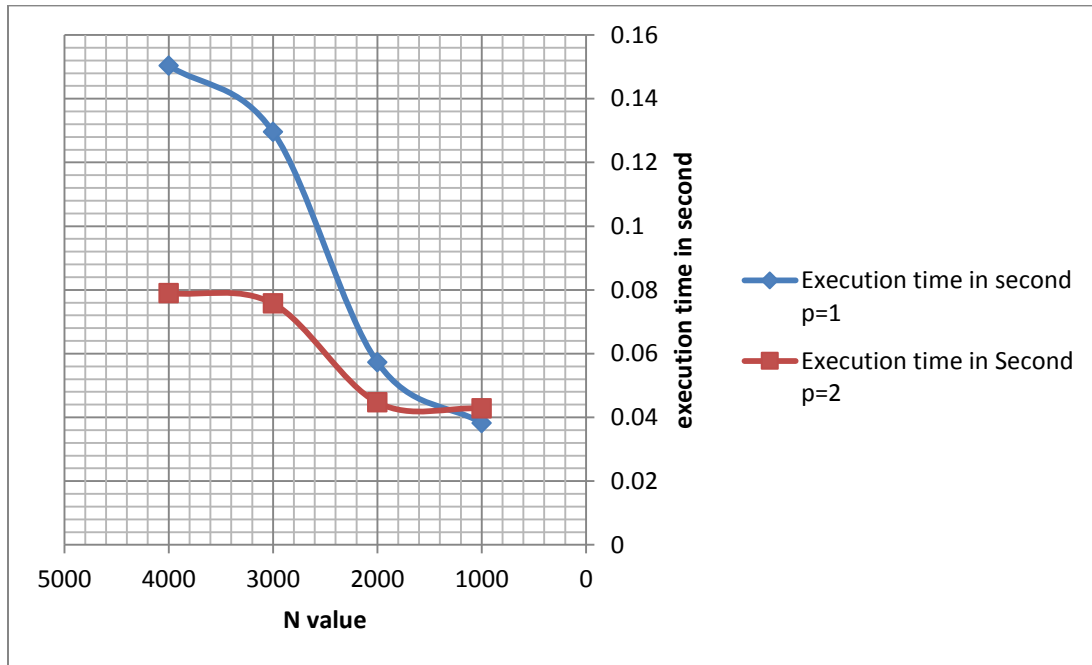


Figure 1.30 show execution time of p1 and p2. N is increasing time is increasing.

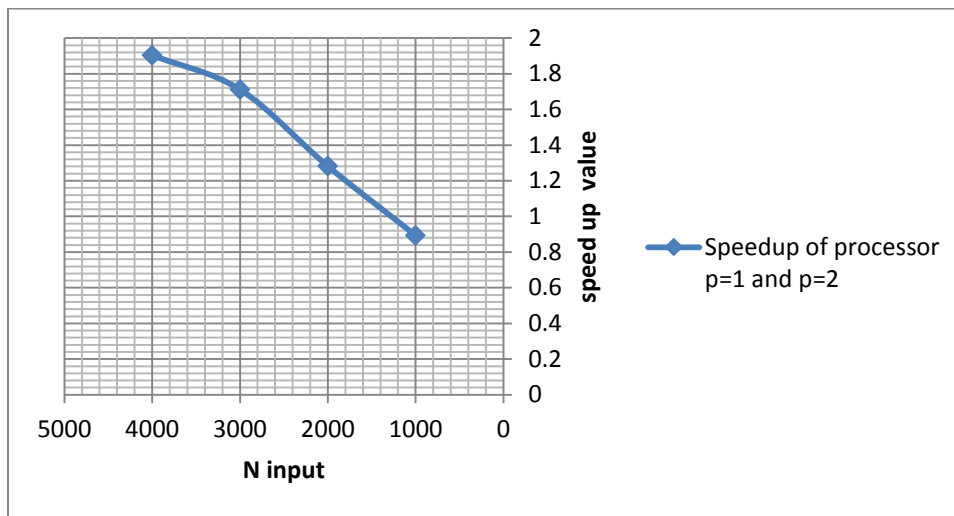


Figure 1.31 speed up of p=1 and p=2 value it approach to near the 2 value

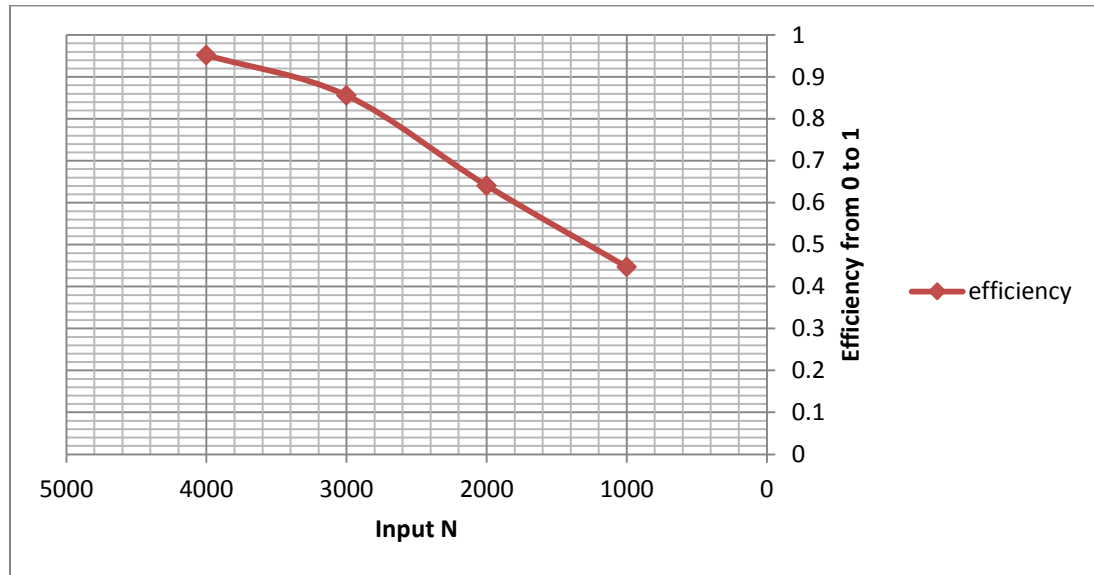


Figure 1.32 this show the efficiency of processors when N increase efficiency increase and going to near the 1 value. Above result show that when our program run on two pc its task is divided into two pc and speed of execution become near to half but not half because of some attenuation using wire and other factors.

12. Conclusion:

In this work we attempted to develop a simple computer program acts as a parallel processing and we get that parallel processing consume less time as compare to sequential time for complex and large tasks. Parallel processing is actually using the resources of system concurrently. Users who want to save the time so parallel processing is efficient way to solve complex tasks in limited time. Now a day increasing the demand of complexity so the parallel processing is major resource to solve these problems.

13. References:-

- 1- http://swash.sourceforge.net/online_doc/swashimp/node9.html. (date: 18/5/2014)
- 2- <http://www.mpich.org/downloads/>.(date: 20/4/2014)
- 3- <http://mpi.deino.net/index.htm> (date: 20/5/2014)
- 4- http://www.scl.ameslab.gov/Projects/mpi_introduction/para_mpi.html