**Assignments for High Performance Computing (CS301) Course/ Lab**

This document will be updated regularly. All assignments have to be submitted by a given **deadline**.

**Submission of assignments**: Through Moodle. TAs will also explain in the Lab.

**Make your report brief and to the point; highlight the 10 important issues listed below (observations + supporting explanations).**

**All the assignments** have to be supplemented with a **brief write-up or ppt with the following details** (wherever necessary):

1. Context:

* Brief description of the problem.
* Complexity of the algorithm (serial).
* Possible speedup (theoretical).
* Profiling information (e.g. gprof). Serial time.
* Optimization strategy.
* Problems faced in parallelization and possible solutions.

1. Hardware details: CPU model, memory information details, no of cores, compiler, optimization flags if used, precision used.
2. Input parameters. Output. Make sure results from serial and parallel are same.
3. Parallel overhead time. (openmp version on 1 core vs serial without openmp)
4. Problem Size vs Time (Serial, parallel) **curve**. Speedup curve. Observations and comments about the results.
5. If more than one implementation, curves for all algorithms in the same plot.
6. Wherever necessary use log scale and auxiliary units.
7. Problem size vs. speedup curve.
8. No. of cores vs. speedup curve for a couple of problem sizes.
9. Measure performance in MFLOPS/sec.

Assignments include submission of codes [optimized serial and parallel codes (multiple versions if applicable) with necessary comments inside the code].

These codes will be useful for the CS courses in coming semesters (preserve). Register at <http://courses.daiict.ac.in>.

**Evaluation** will be based on the **submission of materials (report + codes) (or/and presentation in class).** Date-wise presentation details will be updated in the **list-of-presentation** file at-least one day in advance.

**Assignment 1 (9th Aug) Deadline: 16th August**

1. Take any of your code (with some reasonable no. of functions and subroutines) and do the profiling as discussed in the class. Understand the output. Nothing to be included in the report on this exercise.
2. Use timer function for measuring elapsed time. Figure out what is the best available function for serial and Openmp. Nothing to be included in the report on this exercise.
3. Write a parallel code (openMP) for integration using trapezoidal rule. Serial code was discussed in the class.

Use the parallel code to calculate PI (lecture 5, slide 29) and verify the implementation. In addition to 10 points discussed above, make comments about your observations and the most optimized implementation.

Start with a naïve implementation, and gradually improve it as discussed during the class (using critical, reduction etc.).

Report about the min. no changes required to convert the serial code to parallel.

Measure performance in MFLOPS/sec.

**Assignment 2 (16th Aug) Deadline: 27th August**

Write a **serial code** and **parallel code** for the following (using openMP) (discussed during today’s lecture).:

1. Calculation of pi using series (take large value of “N” for summation)
2. Multiplication of Two vectors followed by summation. (serial and parallel comparison for accuracy of results)
3. Matrix Multiplication (starting with simple implementation – gradually implement Block algorithm as discussed during the lecture, create n x n matrix; take n in multiples of 2 while increasing the size of matrix (go up to 528 and 1024); change size of block to see the effect.)

In addition to 10 points discussed above, make comments about your observations and the most optimized implementation.

**Assignment 3 (23th Aug) Deadline: 13th September**

1. Write a serial code for twist transformation/ image warping of an image as discussed in the class (lecture-10, case study-3). Write a parallel version of this using openMP. Change the image dimension (same image) and compare the serial vs. parallel implementation.

# Image Format (recommended 🡪 PPM an acronym derived from "Portable Pixel Map". Take help of TAs if you cannot figure out the following within an hour)

*You can create own input images by exporting image into PPM images. The easiest way to create image is via external tools (*[*read the link*](http://www.scratchapixel.com/old/lessons/2d-image-processing/reading-and-writing-images-a-simple-image-class/reading-and-writing-images-a-simple-image-class/)*). On Linux, look at “bmptoppm” and “convert” for conversions.*

b) **Filtering:** Consider an image neighborhood surrounding each pixel is defined, and the median value (filter in this case) of this neighborhood is calculated and is used to replace the original pixel in the output image by using the following filter:

Imed [x,y] = median (Iorig [i, j] ; i, j belongs to nbor [x,y])

If you choose a square neighborhood around each pixel, defined using the half-width of the neighborhood, i.e., for a half-width of n, the number of pixels in the neighborhood would be (2n+1)^2. Any neighbors that lie outside the image domain are assigned to be that of the nearest pixel within the image boundary for calculation of median.

To start with - choose a half-width of 3 and an image of 256 x256.

Choose by yourself the important dimensions (which you feel will impact the overall speedup; e.g. image size, filter size) and show the effect of these dimensions on performance. Show your observations using plots, and support you observations with proper explanations.

In addition to 10 points discussed above, make comments about your observations and the most optimized implementation.

**Assignment 4 (13th September) Deadline: 23rd September**

Write a **serial code** and **parallel code** for the following (using openMP):

·         Calculation of pi using random numbers.

·         Reduction.

In addition to 10 points discussed above, make comments about your observations and the most optimized implementation.

First use the random function of the C library for random number generations and then understand the problem of parallel random number generations using OpenMP, and what are the main issues. Report on it briefly.

**Assignment 5 (27th September) Deadline: 28th October**

Compute the inclusive prefix sum (scan) from an input array; store results in output array for different lengths of N (size of the input array). Convert the same code into an exclusive scan (e.g. option-1 – inclusive scan; option-2 exclusive scan).

Develop a parallel version of scan using openMP. (hint: serial- Work: *O*(*n*), Span: *O*(*n*); parallel - Work: *O*(*n*), Span: 2**log** *n* )

**Problem [**efficient parallel filter**] (**do the following with both serial and parallel implementation using the above implementations):

Given an array **input**, produce an array **output** containing **only elements** such that **filter(element)** is **true**

Example: **input [17, 4, 6, 8, 11, 5, 13, 19, 0, 24]**

**filter: is element > 10**

**output [17, 11, 13, 19, 24]**

Choose an Input with length (>10^6); fill the vector with random numbers (integer) between 0-100, and do the above operation.

Things to note

– Finding elements for the output

– And also getting them in the right place.

In addition to 9 points discussed above, make comments about your observations and the most optimized implementation.

**Assignment 6 (8th November) Deadline: 16th November**

(i) Write a parallel code (using MPI directives) for integration using trapezoidal rule. Serial code and OpenMP code has been already implemented.

Use the MPI version to calculate PI and verify the implementation. **(take enough number of trapezoids to study the effect of speedup)**

Run the MPI version on 4, 8 and 16 cores and compare the timings with a serial, and OpenMP version for same accuracy of PI. Which is the best implementation and what are the important observations.

**The following exercise (not to be reported in assignments) will help you n understanding MPI and the related procedure in good details (you can do the following using 2 lab desktops for understanding)**

**--------------------------------------------------------------------------------------------------------------------**

Using “ssh” for communication. To use "ssh", one has to generate a "key".

Now lets take an example.  Suppose, we want to ssh from our machine (machine1) to the machines2,3,4….

The steps we will follow will be this:  
  
1.  generate a "key" by using the command "ssh-keygen"  
  
**$ ssh-keygen -t rsa**  
Generating public/private rsa key pair.  
Enter file in which to save the key (/users/……):<Enter>  
Enter passphrase (empty for no passphrase):<Enter>  
Enter same passphrase again:<Enter>  
Your identification has been saved in .ssh/id\_rsa  
Your public key has been saved in .ssh/id\_rsa.pub.  
The key fingerprint is:  
21:a1:25:58:fb:bd:ee:…..   
  
2. now there are two files which are generated in ".ssh" directory. One is "id\_rsa"(private key) and the other one is "id\_rsa.pub" (public key).  
  
3. create a directory ".ssh" in the remote machine and create a file called "authorized\_keys".  the contents of "authorized\_keys" should be  
same as "id\_rsa.pub" in ur machine.  
  
4. now connect to the remote machine by giving the command "ssh machine??"  
  
Example  
$ ssh machine2  
The authenticity of host 'machine2 (192…..)' can't be established.  
RSA key fingerprint is 79:d9:fb:e4:2e:4d:2a:65: ….  
Are you sure you want to continue connecting (yes/no)? <yes>  
Warning: Permanently added 'machine2,192……..' (RSA) to the list of  
known hosts.  
m…@.....'s password:<enter your password>  
  
$ <this prompt will be the machine2….. prompt>  
  
5.  now the next time you say "ssh machine2", it will log-into machine2 without asking for a password.  if we had entered some passphrase (more than 4 bytes) in step 1 then everytime we do "ssh machine2", it will prompt for entering the "passphrase". hence we do not enter any passphrase while generating the key.  
  
this is the method for one particular machine.  the steps 3-4 has to be done for each machine. the same "public key" has to be copied to  
different machines. it is also possible to have different "public keys" for different machines because everytime "ssh-keygen" is executed it  
generates a different "fingerprint" altogether, but then it becomes more complicated.

Put the machine names in “machines” and execute with

$ mpiexec -n 16 -machinefile machines ./a.out

Put the contents in the “machines” as follows:

ipaddress slots=8

ipaddress slots=8

ipaddress slots=8

Put the executable in all the nodes in the directory with the same path.

**--------------------------------------------------------------------------------------------------------------------**