# Homework 2

1. OpenMP Coding - *not expected to be finished by tomorrow but start wirking on it so it will be of help for the homework problem set 3 on Wednesday!*

   - Start working on a real ("easy" as in having an exact closed form solution) problem - consider the wave equation:

     $$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

     with initial conditions

     $$u(x,0) = \sin(2\pi x), \left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0} = -2\pi c \cos(2\pi x) \tag{2}$$

     on the unit interval $[0,1]$ with periodic boundary conditions $u(0,t) = u(1,t)$. Discretize $u_i^n = u(i\Delta x, n\Delta t)$ and employ the 2nd order centered in space and time finite difference scheme:

     $$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} = c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{3}$$

     to solve equations 1,2 for an arbitrary $c$. $c$, the number of gridpoints $M+1$ (such that $0 \le i \le M$ in the $x-$direction) and number of timesteps $N$ (such that $0 \le n \le N$) ideally should be provided at runtime (by command line options, or by reading a file or through standard input). Keep in mind that $u_0^n = u_M^n$ for all $n$ due to periodicity. The exact solution to compare against is $u(x,t) = \sin(2\pi(x - ct))$.)

     Write a parallel OpenMP version of your program that uses $P$ processors (left to be a runtime variable set via OMP_NUM_THREADS).

     – Try to also do the initialization in parallel.

     – You only need to update using the scheme gridpoints 0 to M-1 as gridpoint M is the same as gridpoint 0.

     – Take care to comply with the CFL condition $c\frac{\Delta t}{\Delta x} \le 1$.

     – To conserve memory you should use an array of arrays (of dimension 3) to store the 3 levels of $u$ needed by the method. You may avoid excessive copying by manipulating the indices to the array, e.g. in Fortran, in each timestep:

```
 DO i=1,M-1
    u(i,new) = dt*dt*c*c*(u(i+1,curr)
$              - 2*u(i,curr)
$              + u(i-1,curr))/(dx*dx)
$              + 2*u(i,curr)
$              - u(i,prev)
 ENDDO
 swap = new
 new = prev
 prev = curr
 curr = swap
```

  Remember than in C/C++ this index variable should be first, not last to ensure that most memory accesses are unit-stride.

  - Every $L$ timesteps (where $L$ is defined at runtime) output the solution, the exact solution as well as the difference from the exact solution to disk (in separate files). Ideally you should also do this at the end of the run.
  - Add appropriate reduction directives to allow for the correct calculation of the maximum and the root-mean-square error to stardard output every $L$ timesteps.

- Change of Boundary and Initial Conditions: Solve the wave equation 1 with initial conditions

$$u(x,0) = \sin(2\pi x), \left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0} = 0 \tag{4}$$

  on the unit interval $[0,1]$ with Dirichlet boundary conditions

$$u(0,t) = 0, u(1,t) = 0 \tag{5}$$

  (pined endpoints). Note the difference in initial conditions and more importantly in boundary conditions. Modify your existing parallel program that solves this using scheme (3). Use separate subroutines for initialization and error calculations etc. so that you can reuse as much of your code as possible (employ orphaned directives).

  - The exact solution to compare against is $u(x,t) = \sin(2\pi x)\cos(2\pi ct)$.)
  - You only need to update gridpoints 1 to M-1.

2. Food for thought

   - Modify (or at least think how you would modify) your program so that it solves a modified Equation (1) in the unit square $[0,1]^2$. The twist is that instead of a constant $c$ we have $c(y) = 1 + 0.1 sin(2\pi y)$ with the Dirichlet boundary conditions $u(0,y,t) = 0, u(1,y,t) = 0$ in the x-direction and periodic boundary conditions $u(x,0,t) = u(x,1,t)$ in the y-direction with initial conditions

     $$u(x,y,0) = \sin(2\pi x)\sin(2\pi y), \left.\frac{\partial u(x,y,t)}{\partial t}\right|_{t=0} = 0 \qquad (6)$$

     and an exact solution of $u(x,t) = \sin(2\pi x)\sin(2\pi y)\cos(2\pi c(y)t)$. The number of gridpoints in the x- and y-directions are supplied at runtime.

     – Is there a preferred loop order to represent the $x-$ and $y-$ directions?

     – Can we use nested parallel loops to parallelize the solution in both the x-loop and y-loop level?

3. Vital MPI preliminaries

   - For those of you that did not do it already, please becoming familiar with the MPI compilation and runtime environment:
   - Write, compile and run the "Hello World" program described in class.
   - Download simple-mpi.tar.gz from the Stellar website (alternatively http://web.mit.edu/13.715/www/simple-mpi.tar.gz), compile and run the examples. Look up MPI_Get_processor_name() and MPI_Get_version() routines in the man pages. Read the code, and produce output for 2 or more processors.

4. Write your own ping-pong program. Employ MPI_Send()/MPI_Recv() and MPI_Wtime() (look at man page) to measure roundtrip time $RTT(s)$ and bandwidth $BWD(s) = 2\frac{s}{RTT(s)}$ as a function of message length $s$. Produce graphs of $0.5RTT(s)$ and $BWD(s)$ against $s$. Report the following 3 values:

   - Define zero-byte latency as $0.5\lim_{s\to 0} RTT(s)$
   - Define asymptotic bandwidth as $\lim_{s\to\infty} BWD(s)$
   - Define maximum bandwidth as $\max_{s\in(0,\infty)} BWD(s)$. In practice $\infty$ is in the range of a few MBs, depending on the speed of the interconnect.

5. Bonus for the Eager Beaver (EB):

   - Calculate the three aforementioned quantities for the three combinations of: MPI_Send() vs MPI_Ssend() vs MPI_Bsend() coupled with MPI_Recv(). Do you need to do anything to guarantee correctness with MPI_Bsend()?
   - Since your nodes are dual processor nodes: Try two processors on the same machine vs two processors on different machines. (Can you force the MPI runtime to do this? Read the manpages for mpirun etc. for OpenMPI very carefully.) Try with the other three variants of MPI_Send() as well.
   - Try timing a streaming ping, where process A continuously streams data to process B. How to best time this?
   - What problems, if any, do you encounter? What are your findings?