



# Large-Scale Simulations on Parallel Computers

Grétar Tryggvason  
Spring 2010



## Outline

- Basic Machine configurations
- Parallelization
- The Message Passing Interface (MPI) library

Parallel computing has become the way to increase computer power. All of the worlds fastest computers are massively parallel (see <http://www.top500.org>) and parallel computers are becoming common in industry and at universities.



## Machine configurations

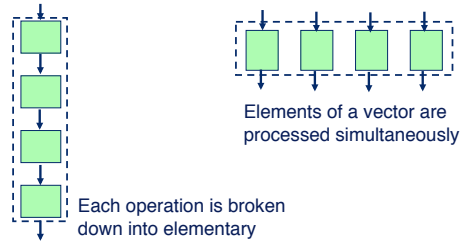
SISD: Single Instruction, Single Data

SIMD: Single Instruction, Multiple Data

MIMD: Multiple Instruction, Multiple Data

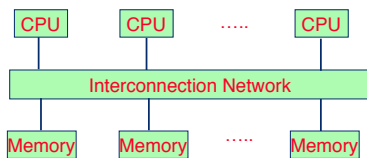


## Early SIMD: Pipeline and Vector Architectures



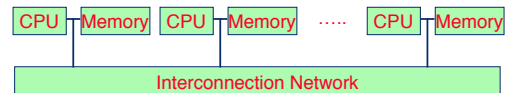
## MIMD: Multiple Instruction, Multiple Data

### Shared Memory MIMD



## MIMD: Multiple Instruction, Multiple Data

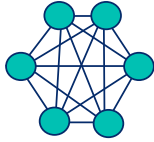
### Distributed Memory MIMD



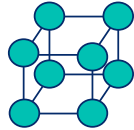


MIMD: Multiple Instruction, Multiple Data

Distributed Memory MIMD: Static Interconnection networks



Fully connected interconnection network



3D hypercube



The interconnection network can be either static or dynamic and different vendors have elected to go with different configurations.



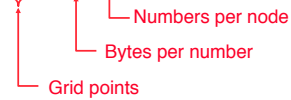
### Parallelizing the solution to the heat equation An Example



$$\frac{\partial f}{\partial t} = \alpha \left( \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)$$

Assume we want to solve this on a very large grid, say 20,000 by 20,000. The storage requirement is:

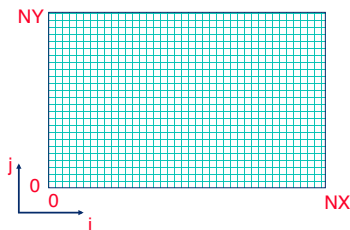
$$4 \times 10^8 \times 8 \times 2 = 6.4 \text{ GB}$$



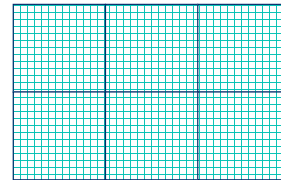
It used to be unlikely that this would fit into the RAM of one node and we therefore had to divide the problem between several processors



Domain Decomposition

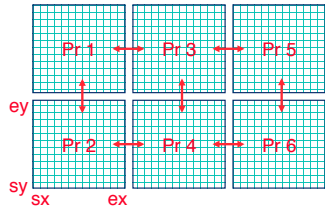


Domain Decomposition

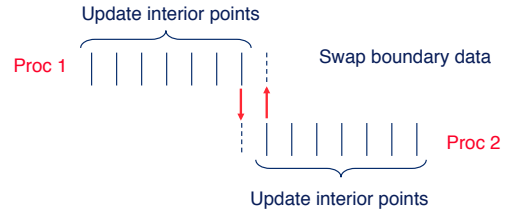




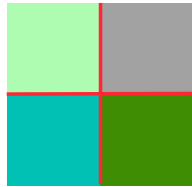
Domain Decomposition



Usually the grids must overlap



For 2D and 3D problems it is necessary to exchange lines and planes of data



Each processor updates the portion of the grid that resides on that node and swaps data with its neighbors



```

c Advance f in time
subroutine adv_in_time( f, fo,b,sx,ex,sy,ey,h,dt)
integer sx,ex,sy,ey,i,j
double precision f(sx-1:ex+1,sy-1:ey+1), fo(sx-1:ex+1,sy-1:ey+1),
& b(sx-1:ex+1,sy-1:ey+1)
do 10 j=sy, ey
do 10 i=sx, ex
fo(i,j) = f(i,j)
10 continue
do 10 j=sy, ey
do 10 i=sx, ex
f(i,j) = fo(i,j) + (dt/h*h)* (fo(i-1,j)+fo(i,j+1)+fo(i,j-1)+fo(i+1,j)
- 4.0*fo(i,j)) - h*h*b(i,j)
10 continue
return
end

```



For the complete problem we must determine:

How the processors are connected

How to prepare the data to be sent if it is not continuous in memory

How to transfer the data in the right order so the data is there when needed



In Message Passing, the processors explicitly send and receive data

Generic form:

Send(address, length, destination, tag)

Receive(address, length, source, tag, actlen)

actlen: length of the message received

In actual implementations the arguments are slightly more complex, due to the need to deal with data that is not continuous in memory



The structure of the program

```

Initialize parallelization
Determine setup, including number
of processors and connectivity
If master, determine size
Broadcast parameters
Do itime=1,MaxSteps
    Swap data
    Advance f
end do
Gather data/print
End parallelization

```



Measuring how successful the parallelization is:

$$\text{Speedup} = \frac{\text{Time for 1 processor}}{\text{Time for } p \text{ processor}}$$

Perfect speedup

$$\text{Speedup} = \frac{T}{T/n} = n$$

Usually the performance degrades as n increases



## MPI: Message Passing Interface



What it is:

- MPI is a library, not a language. It consists of subroutines that are called from FORTRAN, C, or C++ programs to facilitate parallelization of programs
- MPI is a specification, not a particular implementation. All parallel computer vendors currently offer MPI implementation for their machines
- MPI is designed for the message passing model



General structure of an MPI parallel program

```

program main
include "mpif.h"
call MPI_INIT( ierr )
call MPI_COMM_RANK(MPI_COMM_WORLD, myid, ierr )
call MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr )

```

Actual code

```

call MPI_FINALIZE(ierr)

```

Identity of  
each processor

Initialize MPI:

Number of  
Processor  
used



Run the same program on all processors.  
Set up problem on master processor by:

```

if ( myid .eq. 0 ) then
    Set up problem
end

```

The other processors are referred to as  
"slaves" or "workers"



Exchanging data

Data transfer needs to be synchronized

Blocking Send

Ordered Send

Sendrecv

Buffered Send

Nonblock lsend



Send information to all processors (broadcast)

```
call MPI_BCAST(data, brows, MPI_DOUBLE_PRECISION,
& master, MPI_COMM_WORLD, ierr)
```

Send and receive information

```
call MPI_SENDRECV(a(sx,ey),nx,MPI_DOUBLE_PRECISION,
& nbrtop, 0,
& a(sx,sy-1), nx, MPI_DOUBLE_PRECISION,
& nbrbottom, 0, comm2d, status, ierr )
```



Set up a "communicator" for a cartesian grid:

c Get a new communicator for a decomposition of the domain.  
c Let MPI find a "good" decomposition

```
c
  dims(1) = 0
  dims(2) = 0
  call MPI_DIMS_CREATE( numprocs, 2, dims, ierr )
  call MPI_CART_CREATE( MPI_COMM_WORLD, 2, dims,
* periods, .true.,comm2d, ierr )
```

c Get my position in this communicator

```
c
  call MPI_COMM_RANK( comm2d, myid, ierr )
```



Timing the program

```
t1 = MPI_WTIME()
```

```
t2 = MPI_WTIME()
```

Several graphics based program can be used to analyze the performance of the code

Derived datatypes:

When data is not continuous in memory, MPI allows us to set up derived datatype. The simplest one is when we send every nth element of a vector



### Using MPI: Portable Parallel Programming with the Message-Passing Interface

by [William Gropp](#), [Ewing Lusk](#), and [Anthony Skjellum](#)  
Published in 1999 by [MIT Press](#), 371 pages.

<http://www-unix.mcs.anl.gov/mpi/index.html>



## ME612 Computational Fluid Dynamics Summary

Grétar Tryggvason  
Spring 2010



- What did we learn?
- Coarse goals and brief outline
- More detailed list of topics
- Grading
- Exam
- Questionnaire/input



**Coarse Goals:**

Learn how to solve the Navier-Stokes and Euler equations for engineering problems.

Hear about various concepts to allow continuing studies of the literature.

**Ways:**

Detailed coverage of selected topics, such as: simple finite difference methods, accuracy, stability, etc.

Rapid coverage of other topics, such as: multigrid, monotone advection, unstructured grids.



**Part I**

A Two Week introduction to CFD, including an introduction to a commercial package (FLUENT)

**Part II**

Standard Numerical Analysis of partial differential equations, cumulating in solution techniques for the Navier-Stokes equations

**Part III**

Advanced topics in CFD



Introduction, what is CFD, examples, computers, elementary numerical analysis, course administration

Elementary numerical analysis, integration of ordinary differential equations

Elementary numerical analysis, accuracy, stability, partial differential equations

Review of fluid mechanics: the governing equations

Commercial solvers



First order Partial Differential Equations (PDF's). Characteristics. Classification of Second Order PDF's.

Algorithms for Hyperbolic equations. The Euler equations.

Algorithms for Parabolic equations.

Algorithms for Elliptic equations.

Finite Difference solution of the Navier-Stokes Equations in vorticity/streamfunction form

Finite Volume Approach, Solving the Navier-Stokes Equations in Primitive Variables, the MAC Method



Complex Domains. Body fitted Coordinates.

Complex Domains. Grid Generation

Introduction to Turbulence, Multiphase flow, and Combustion

Parallel Computations, Visualization

Direct Numerical Simulations of Multiphase Flows



**Project 1**

Warm-up. Solving a one-dimensional unsteady advection-diffusion problem

**Project 2**

Flow in an L-shaped channel using FLUENT

**Project 3**

One dimensional shock capturing using Lax-Wendroff method

**Project 4**

Modify a code for the Navier-Stokes equations in the primitive variables (pressure and velocity)



**The Navier Stokes Equations:**

The basic equations of fluid mechanics in integral and differential form. The difference between a conservative and non-conservative form. Different formulations of the basic equations: Primitive (velocity-pressure) form; stream function-vorticity, and velocity-vorticity form in two- and three-dimensions. The representation of advection, viscous friction, and incompressibility, in the equations of motion. The pressure equation for the primitive formulation. Physical boundary conditions.



**Partial Differential Equations:**

Characteristics and the basic theory of first order partial differential equations. Second order partial differential equations and how to convert them to a system of first order equations. Hyperbolic, parabolic and elliptic second order equations and their physical significance. What is an ill-posed problem. Conservation laws and the conservative form. Classical model equations and their solutions. Diffusion, linear and nonlinear advection, and the Laplace equation. The advection-diffusion equation and its relation to the Navier-Stokes equations.



**Elementary Numerical Concepts:**

Discrete approximations; finite differences; finite volume; spectral and finite elements. Numerical differentiation and integration. Time integration: Euler, Runge-Kutta and predictor-corrector. Implicit time integration.



**Numerical Solutions of PDE's:**

Accuracy by Taylor series expansion. Consistency and the Modified Equation. Stability by the von Neumann's method. Basic finite difference schemes for hyperbolic, parabolic and elliptic equations.

Parabolic equations: The forward in time, centered in space method. Implicit versus explicit methods. Backward Euler's method, ADI and splitting. Stability limits. The Crank-Nicholson method, predictor-corrector schemes.

Elliptic equations: Iterative solutions of elliptic equations. Jacobi, Gauss-Seidel, and SOR iterations. Fundamentals of multigrid methods.



Hyperbolic equations: The method of characteristics. Finite difference methods: upwind, leap-frog and Lax-Wendroff's method. The Courant conditions. Advection of sharp shocks: Numerical diffusion and oscillations. Godunov's theorem, Monotonic advection, artificial viscosity, and high order Godunov methods. Linear versus nonlinear advection.

The advection-diffusion equation. Cell Reynolds numbers and the stabilization of advection methods by diffusion terms.



**Numerical Solutions of the Navier-Stokes equations:**

Vorticity-streamfunction form. Order in which the equations must be solved. Boundary conditions for the vorticity.

Primitive variables. The MAC method and staggered grids. Solution strategies and projection methods. Derivation of a discretized pressure equation and treatment of boundaries. Boundary conditions for the velocities. Higher order methods: Adams-Bashford/Crank-Nicolson, predictor-corrector.



**Complex domains:**

Boundary fitted grids. Derivatives and partial differential equations in the mapped coordinates. One dimensional stretching and algebraic grid generation. Elliptic grid generation and elementary control strategies. The vorticity-streamfunction equation in mapped coordinates and its solution. Alternative strategies: Mapping versus unstructured grids. Rectangular and triangular cells. Block-structured grids.



**Complex flows:**

Additional considerations for stratified and turbulent flows. The  $k-\epsilon$  model for turbulent flows.

Computations of multiphase flows. Lagrangian and Eulerian modeling of disperse flow

Combustion modeling: Diffusion flames versus Premixed flames



**Parallel computing**

Serial versus parallel, Shared versus distributed memory, Domain decomposition, Message Passing Interface (MPI)

**Commercial codes**

Major players (Fluent/ANSYS, StarCD). Solution process: pre-processing and post-processing



Grading

Projects	50%
Homework	25%
Quizzes	25%

Project and HW due May 3.



Quiz 2:

Three questions

1. Multiple choice  
(covering everything)
2. Complex domains
3. Solution to a professional problem

Open book and notes



Homework

Elementary Concepts (problems 1, 4, 5)  
Basic Numerical Analysis (problems 2, 3, 6, 7, 9)  
Partial Differential Equations (problems 8, 10-14)  
Mapped Grids (problems 15, 16)  
Solution Approach (problems 17-20)



What if I don't have MATLAB?

Try OCTAVE (free)

<http://www.octave.org/>