PARALLEL COMPUTING IN CFD

Tony Saad

January 2005

http://tsaad.utsi.edu - tsaad@utsi.edu

TOPICS:::

- The Motivation……..(4 minutes)
- The Details……(2 minutes)
- The Plan……(14 minutes)
- The Outcome……(10 minutes)
- The End……(5 minutes)
- The Questions……..(n minutes)

(35 + n) minutes

The Motivation:::

- Most engineering solutions are obtained through computer simulation
- As computational power increased, the sophistication of designs also increased
- Scientists are craving for larger computational power, faster execution and reliable simulations
- Companies nowadays expect designs to be performed within a reasonable amount of time

Physics is the universe's operating system.
(Steven R Garman)

The Motivation:::

- For example, decisions upon a casting mould have to be made within minutes (not to say seconds!)
- Numerical weather prediction is expected to predict the future, not the past!
- Numerical relativity, quantum mechanics are also part of this list
- ***The faster the solution, the better the design!***

There is simply no substitute for knowing what you're doing
(Jeff Case)

The Motivation:::

- Examples of equations
  - **Conservation of Mass:** $$\frac{\partial \rho}{\partial \tau} + \frac{\partial \rho u_i}{\partial x_i} = 0$$
  - **Conservation of Momentum:**

$$\underbrace{\frac{\partial \rho u_i}{\partial \tau}}_{Accumulation} + \underbrace{\frac{\partial \rho u_j u_i}{\partial x_j}}_{Convection} = \underbrace{\frac{\partial}{\partial x_j}\left(\mu \frac{\partial u_i}{\partial x_j}\right)}_{Diffusion} - \underbrace{\frac{\partial p}{\partial x_i} + \rho g_i + s_i}_{Source}$$

In mathematics you don't understand things. You just get used to them. (John Von Neumann)

The Motivation:::

- Examples of equations
  - **Conservation of Energy:**

$$\frac{\partial \rho h}{\partial \tau} + \frac{\partial \rho u_i h}{\partial x_i} = \frac{\partial}{\partial x_i}\left(k \frac{\partial T}{\partial x_i} \right) + s$$

  - **Conservation of Passive Scalars:**

$$\frac{\partial \rho \phi}{\partial \tau} + \frac{\partial \rho u_i \phi}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\Gamma \frac{\partial \phi}{\partial x_i} \right)$$

The Motivation:::

- The increase in processing power has been doubling every two years
- What next? Will we reach processors with millions of gigaflops?

Not all that counts can be counted;
Not all that can be counted, counts

The Motivation:::

- How can we achieve large computational power?
  - Extremely fast processors (millions of teraflops)
  - Extremely large memories
- And that would be it… or not?
- In fact, this could not be the case

Learning is ever in the freshness of its youth, even for the old (Aeschilus)

The Motivation:::

- There are two fundamental limiting factors that do not allow this to happen
  - The speed of light
  - The heat dissipation from CPU's & memory chips
- One promising alternative (the only choice!) is ***parallel computing***…

Everything that you could possibly imagine, you will find that nature has been there before you
(John Berril)

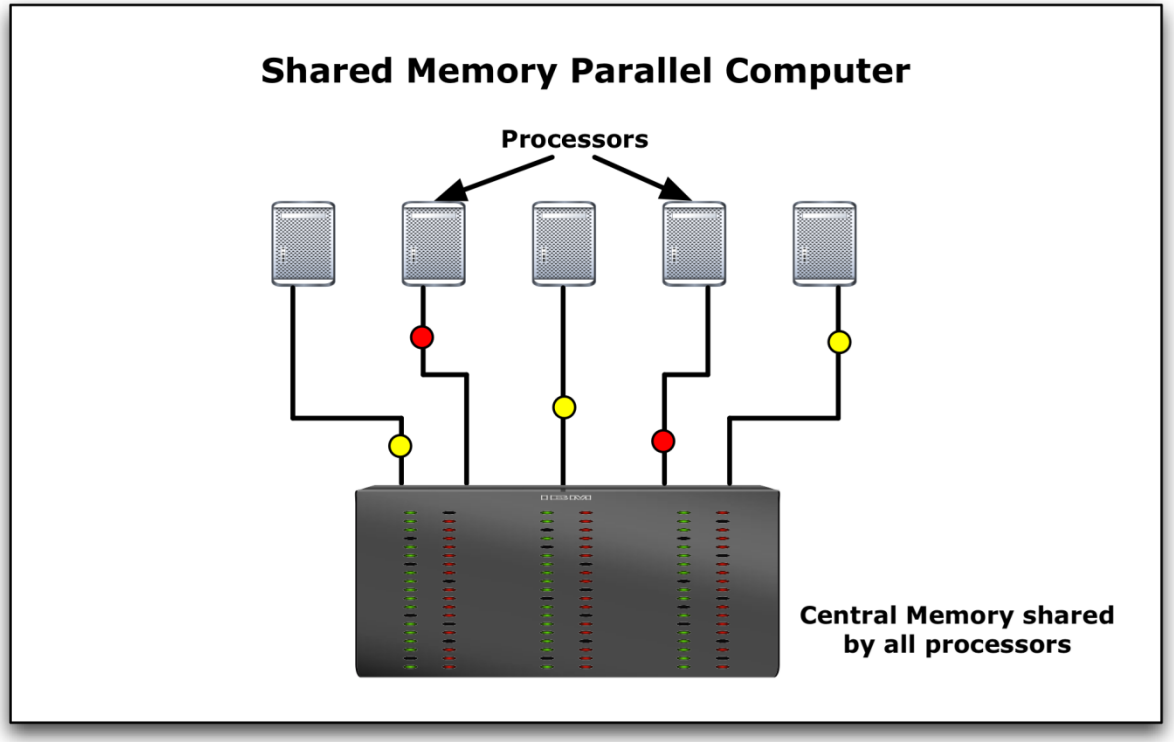
The Details:::

- **Parallel Computing** *stands for the collaboration of several independent processors in the solution of a given problem*
- There are two fundamental types of parallel computers:
  - Shared memory parallel computers
  - Distributed memory parallel computers

The most incomprehensible thing about the world is that it is comprehensible (Albert Einstein)

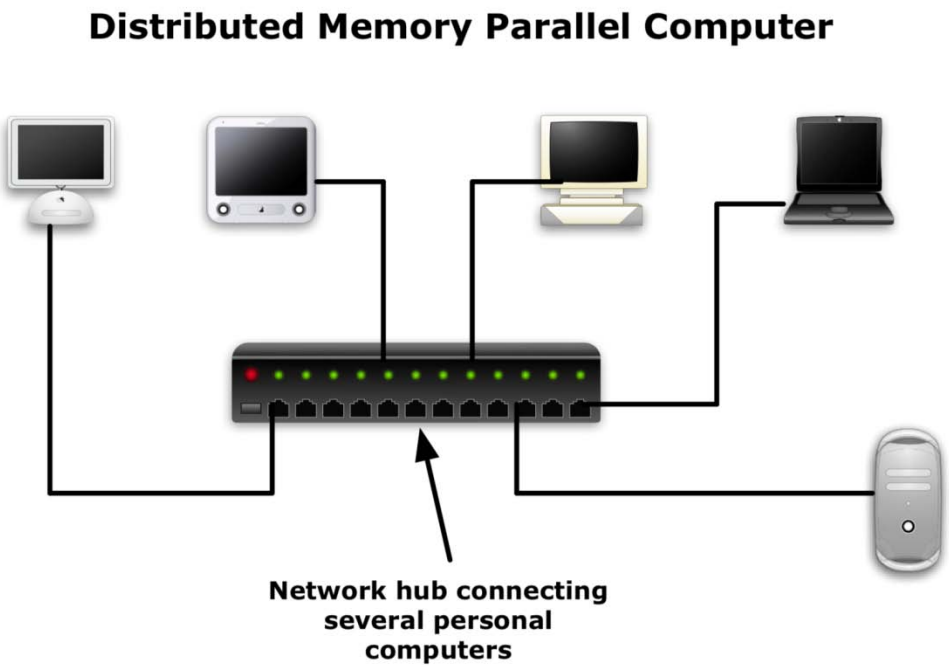
The Details:::

- **Shared Memory**: all processors share a central memory accessible by



The Details:::

- **Distributed Memory:** Each processor has its own local memory. Processors are connected via a network switch



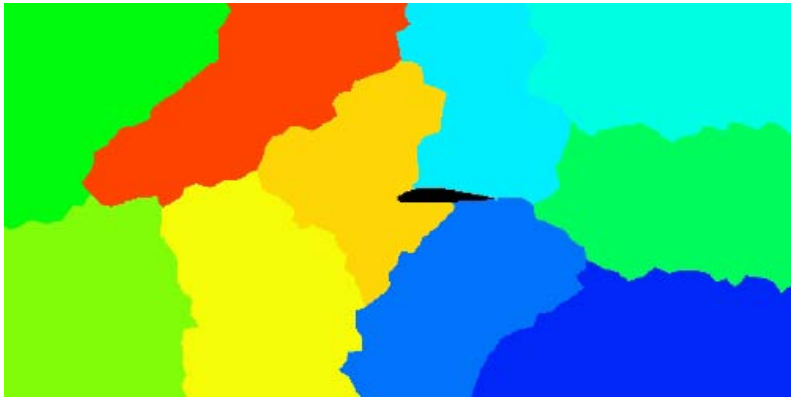
The Plan:::

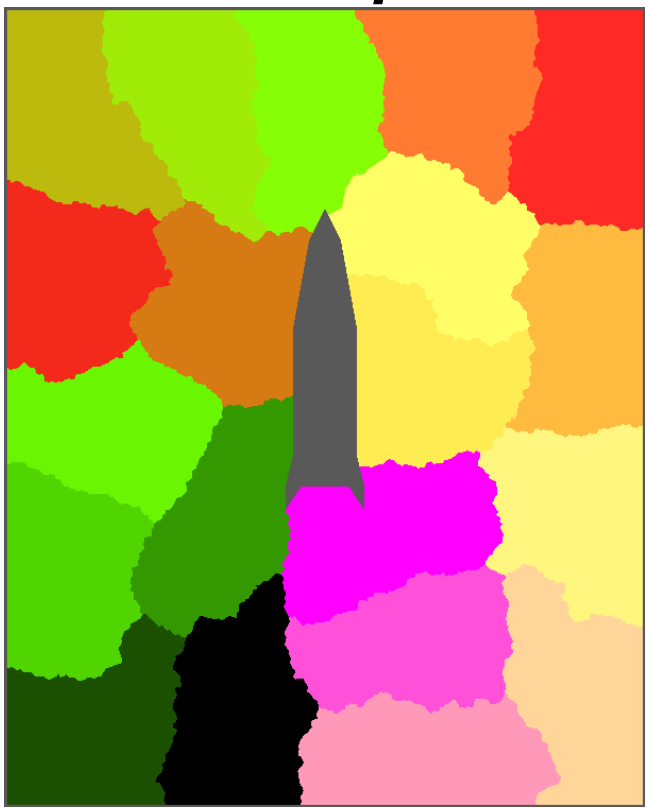
- There are ***FIVE*** conceptual steps in order to successfully parallelize a CFD solver
  1. Domain Partitioning
  2. Partition re-indexing
  3. Master-Node communication of partitions
  4. Parallelization of the linear solver
  5. Node-Master communication of the solution

The plan is nothing; the planning is everything
(David Eisenhower)

I- Domain Partitioning

- As the name Designates, DP (DOMAIN PARTITIONING!) stands for the subdivision of a given ***mesh*** into ***n*** sub-meshes known as ***partitions***.





I- Domain Partitioning

- Each partition will “contain” the same physics (i.e. equations) as the original mesh.
- Partitioning ensures that we have approximately an equal number of elements in each partition
- There are various algorithms to achieve partitioning. Basically, we start by dividing the domain into two, and then each of the sub-domains will be divided into two…and so on…

An expert is a man who has made all the mistakes which can be made in a very narrow field (Niels Bohr)

I- Domain Partitioning

- Why do we partition?
  - Obviously because solving a set of equations on smaller meshes (and then combining the solution) is much faster than solving for the whole mesh (on a single processor)
  - A more important reason for partitioning is the mesh assembly by itself. If we were to do the parallelization on the matrix or loop levels, we would be wasting a lot of time in reassembling the topologies

Research is what I'm doing when I don't know what I'm doing
(Wernher von Braun)

II – Partition Re-Indexing

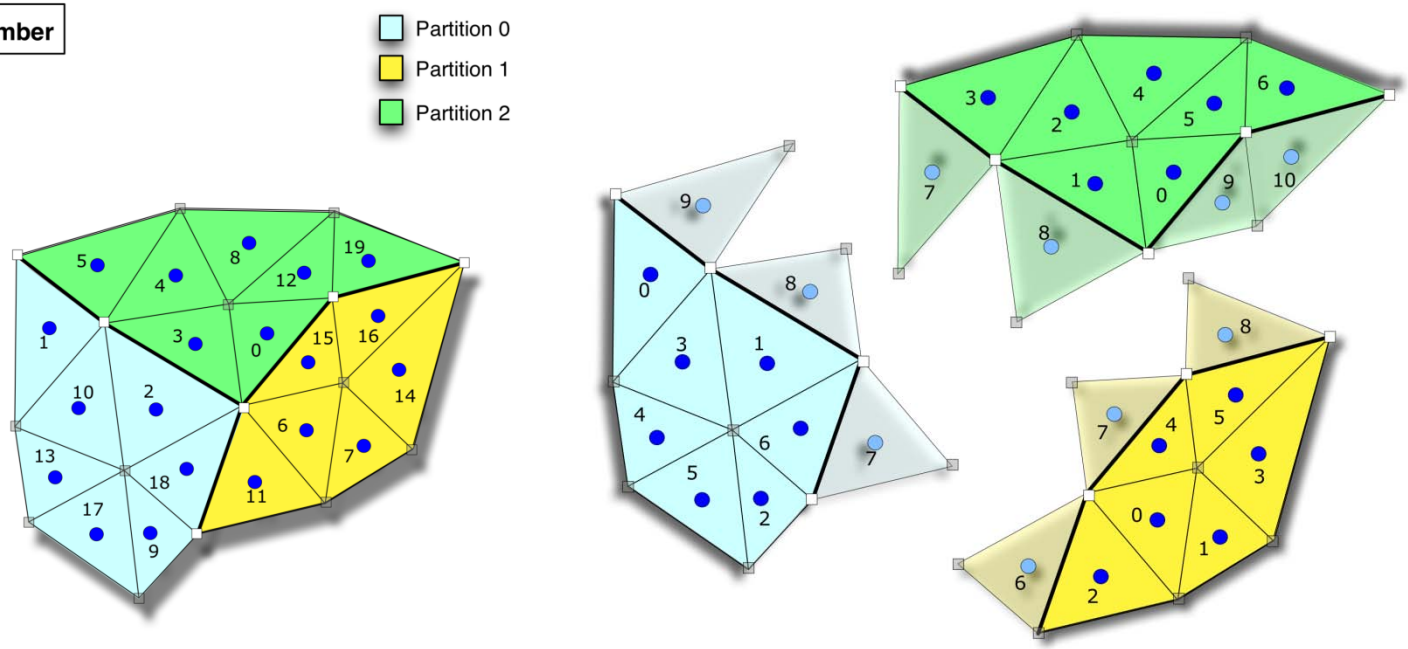
- Once we have generated a set of lovely partitions, we need to set up an independent data structure for each partition.
- In Other words, each partition will be treated as if it were an independent mesh
- This is why we need to re-index the mesh entities
- As mentioned earlier, each of these meshes inherits the properties of the original mesh (such as the equations, boundary conditions etc…)
- This approach confirms the use of the SPMD (single program multiple data) paradigm

A scientist discovers that which exists. An engineer creates that which never was
(Theodore Von Karman)

II – Partition Re-Indexing

Global Cell --- Partition Number

0	2
1	0
2	0
3	2
4	2
5	2
6	1
7	1
8	2
9	0
10	0
11	1
12	2
13	0
14	1
15	1
16	1
17	0
18	0
19	2



Partition 0

Core Local	Core Global	Shadow Local	Shadow Local	[Partition]
0	1	7	2	[1]
1	2	8	1	[2]
2	9	9	3	[2]
3	10			
4	13			
5	17			
6	18			

Partition 1

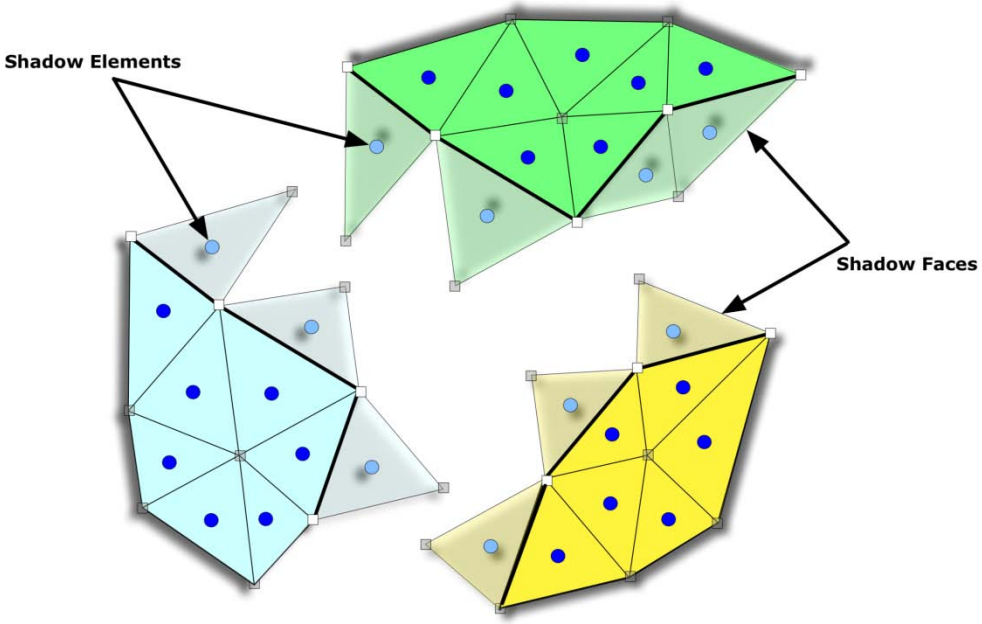
Core Local	Core Global	Shadow Local	Shadow Local	[Partition]
0	6	6	6	[0]
1	7	7	0	[2]
2	11	8	6	[2]
3	14			
4	15			
5	16			

Partition 2

Core Local	Core Global	Shadow Local	Shadow Local	[Partition]
0	0	7	0	[0]
1	3	8	1	[0]
2	4	9	4	[1]
3	5	10	5	[1]
4	8			
5	12			
6	19			

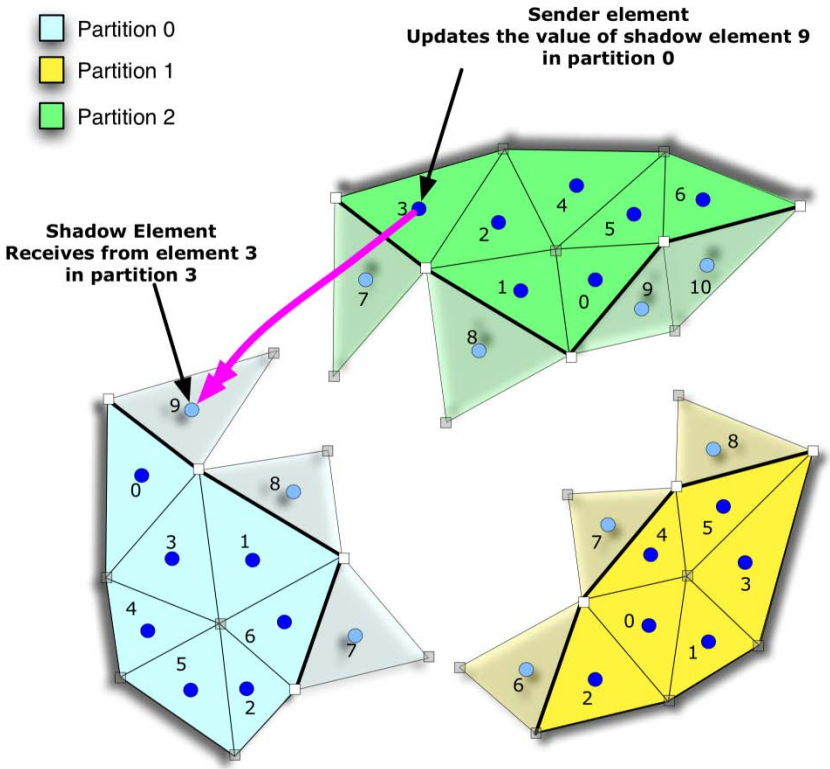
II – Partition Re-Indexing

- At the interface, we need some special treatment
- Shadow entities and Sender entities need to be defined
- A ***shadow entity*** is any topological entity that is not ***core*** to a given partition



II – Partition Re-Indexing

- A ***sender entity*** is any topological entity that is responsible for updating a corresponding ***shadow entity*** in another partition

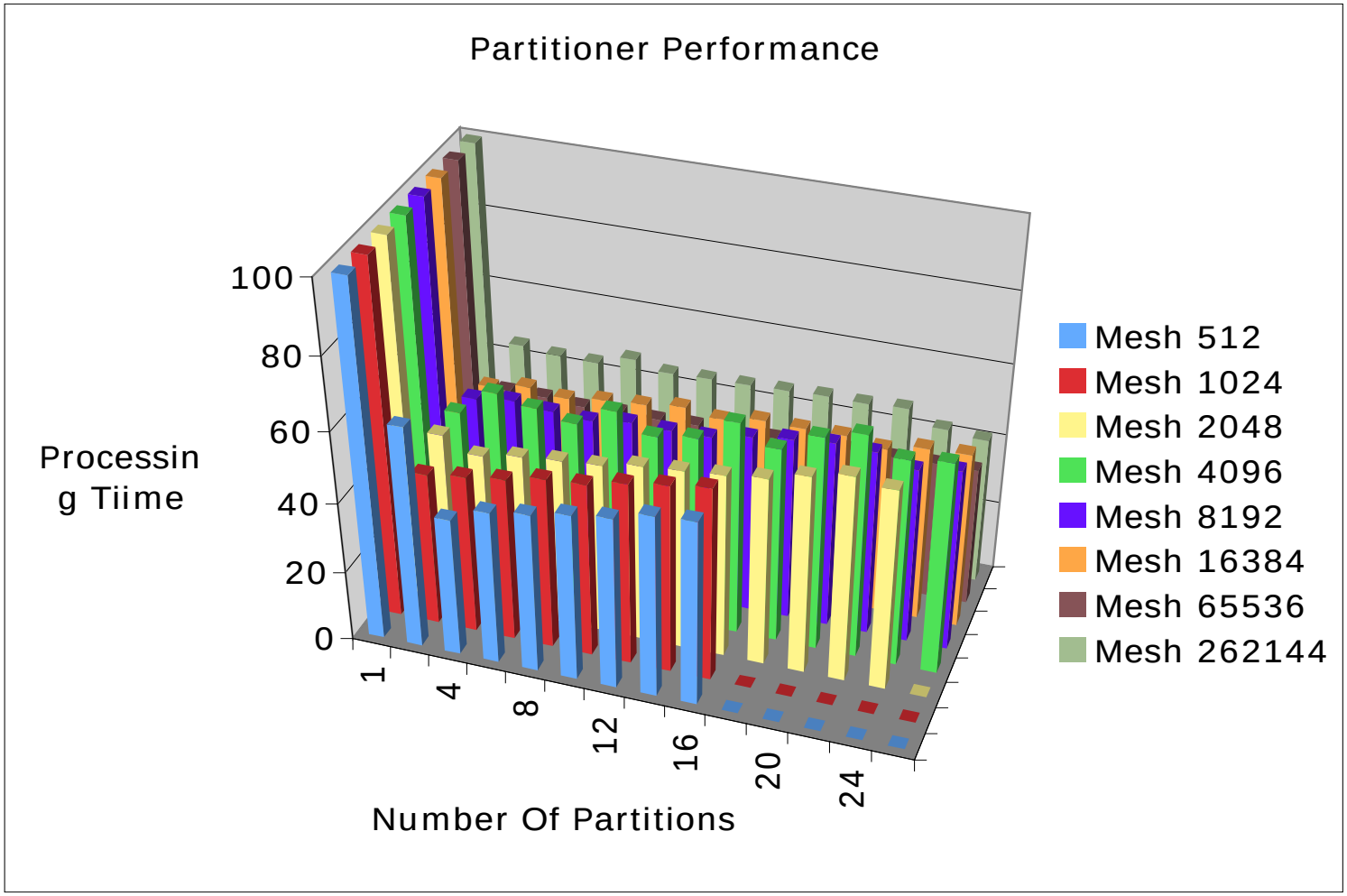


II – Partition Re-Indexing

- Partition re-indexing has to be done for the elements, faces and nodes.
- Each partition should have a knowledge of:
  - its neighboring partitions
  - its shadow entities
  - Its sender entities
- We have implemented a versatile partitioner in an object oriented approach
- The partitioner performs all the above mentioned tasks
- The inner workings of the algorithms used to generate the partition indices are out of scope here

No one is listening until you make a mistake
(Anonymous)

II – Partition Re-Indexing



The only real mistake is the one from which we learn nothing (John Powell)

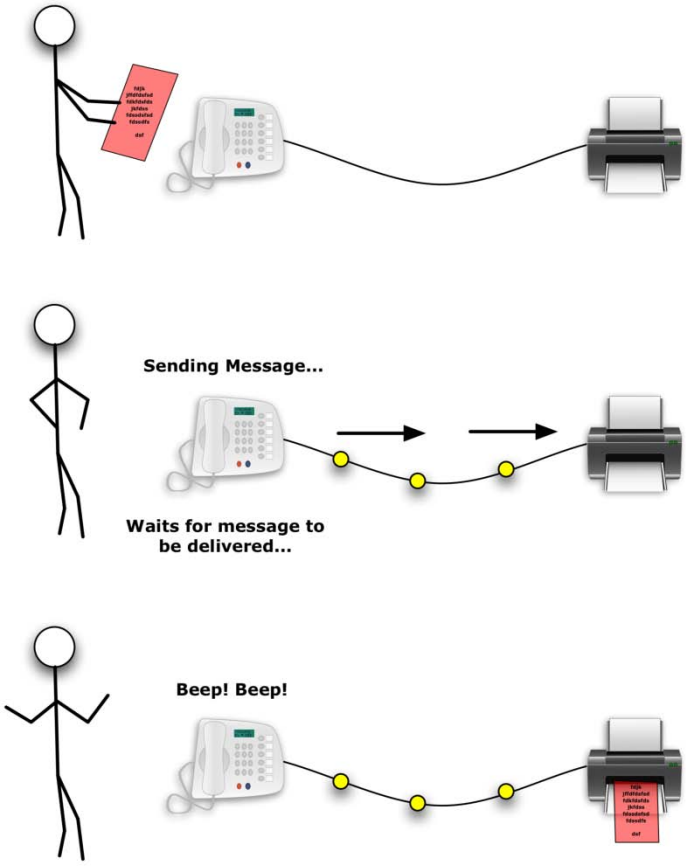
III – MASTER-NODE Communication

- Once we have generated a satisfying partitioning scheme, we need to send the partitions to their corresponding processors
- This is the first encounter with an inter-processor communication in our implementation
- Communication between any two processors (at one time) is known as ***point-to-point (P2P)*** communication
- P2P communication is thus formed by a ***Send*** operation and a ***Receive*** operation

If you tell the truth, you don't have to remember anything (Mark Twain)

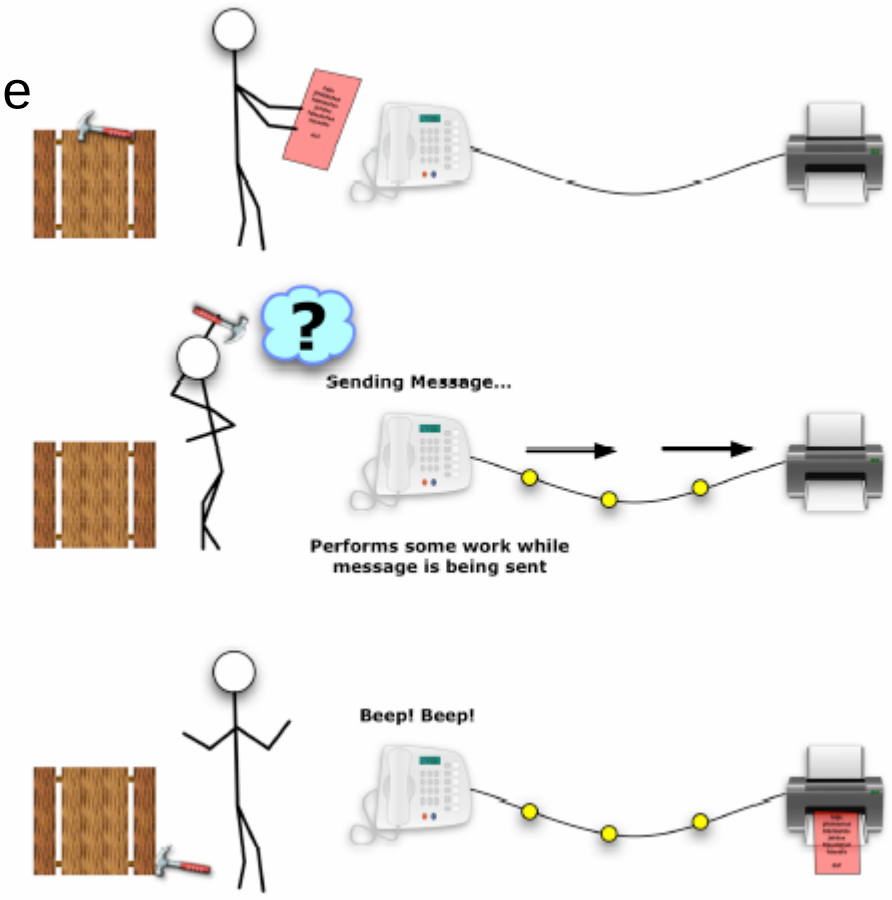
III – MASTER-NODE Communication

- There are two fundamental modes for P2P communication
- **Block-based communication:** The Send does not complete unless it is safe to reclaim the memory allocated in the buffer



III – MASTER-NODE Communication

- **Immediate Communication**: The send completes immediately regardless of the status of the message (whether it was received or not)



III – MASTER-NODE Communication

- Both methods have their advantages & disadvantages
- Block-Based communication makes sure that the message has been safely received, however, if the message is extremely large, buffer memory might not be available
- Immediate communication doesn't worry about buffering (to some extent) however, we are not always certain that the message has been received.
- In this step, we have used block based communication to be extremely certain that the problem definition has been received by all partitions

Who cares how it works, just as long as it gives the right answer?
(Jeff Scholnik)

IV- The Linear Solver

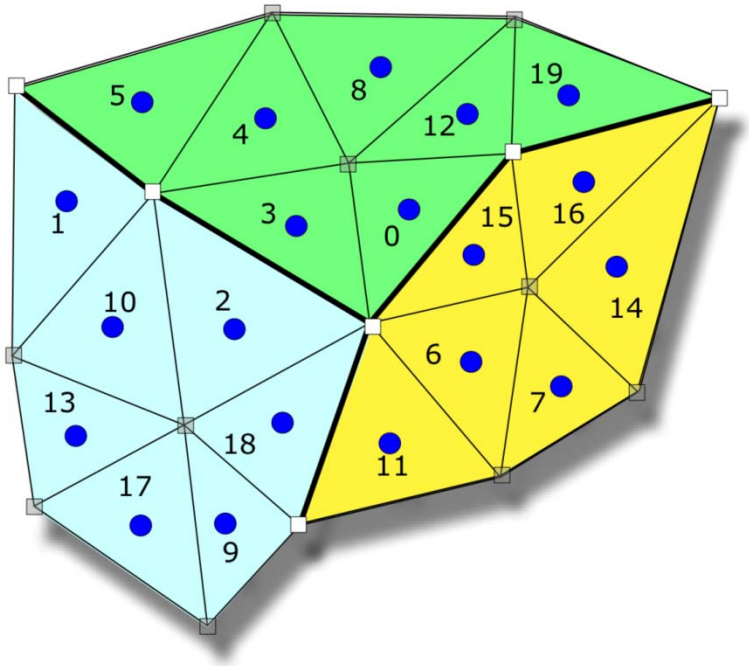
- This is probably the most important step in the parallelization of a CFD code
- A parallel CFD solver requires 3 steps:
  - Solution
  - Communication
  - Decision
- The solution step should be exactly the same as that of the serial solver with one fundamental difference:

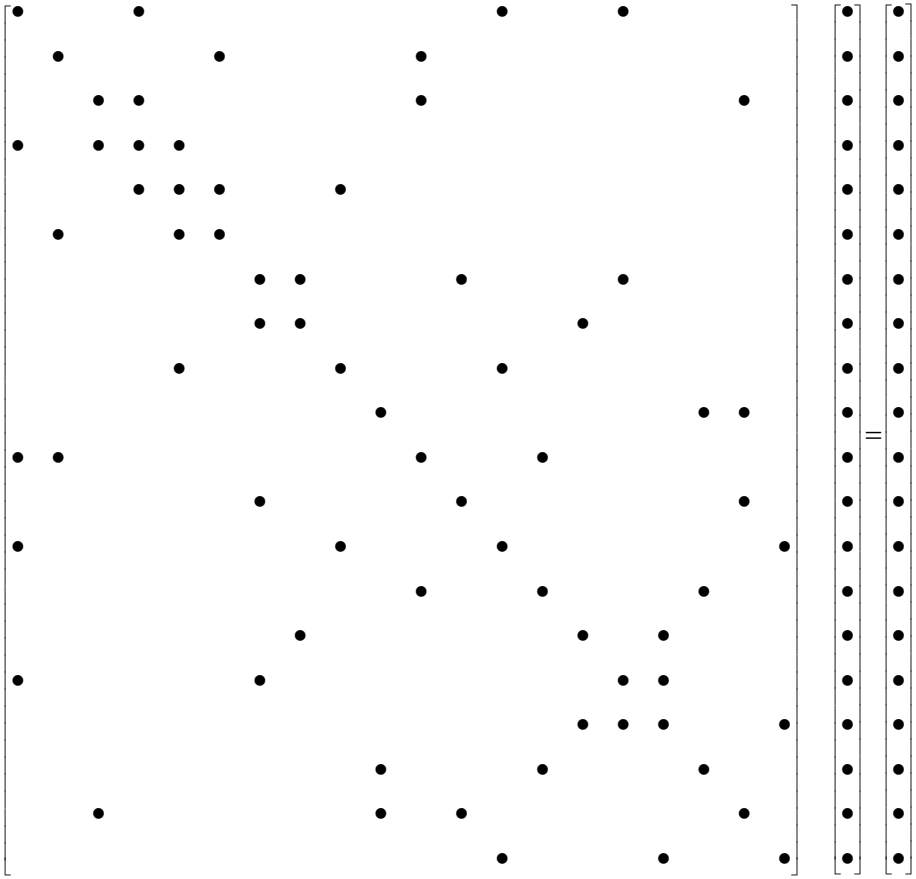
THE SHADOW ELEMENTS!

The reason i saw further than others was that i stood
On the shoulders of giants…
(Sir Isaac Newton)

IV- The Linear Solver

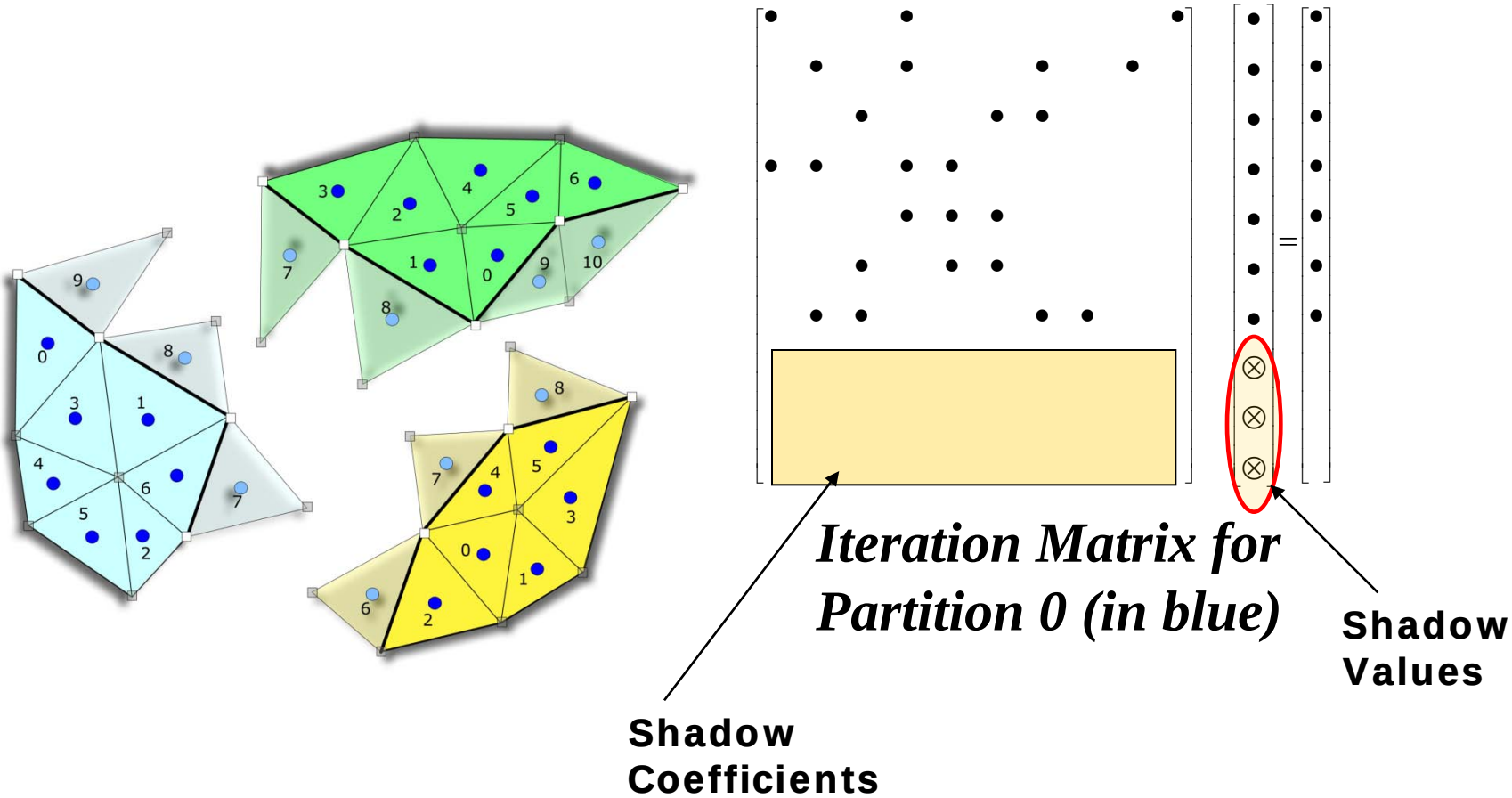
- The iteration matrix of a serial solver looks like this:





IV- The Linear Solver

- The iteration matrix of a certain partition looks like this:

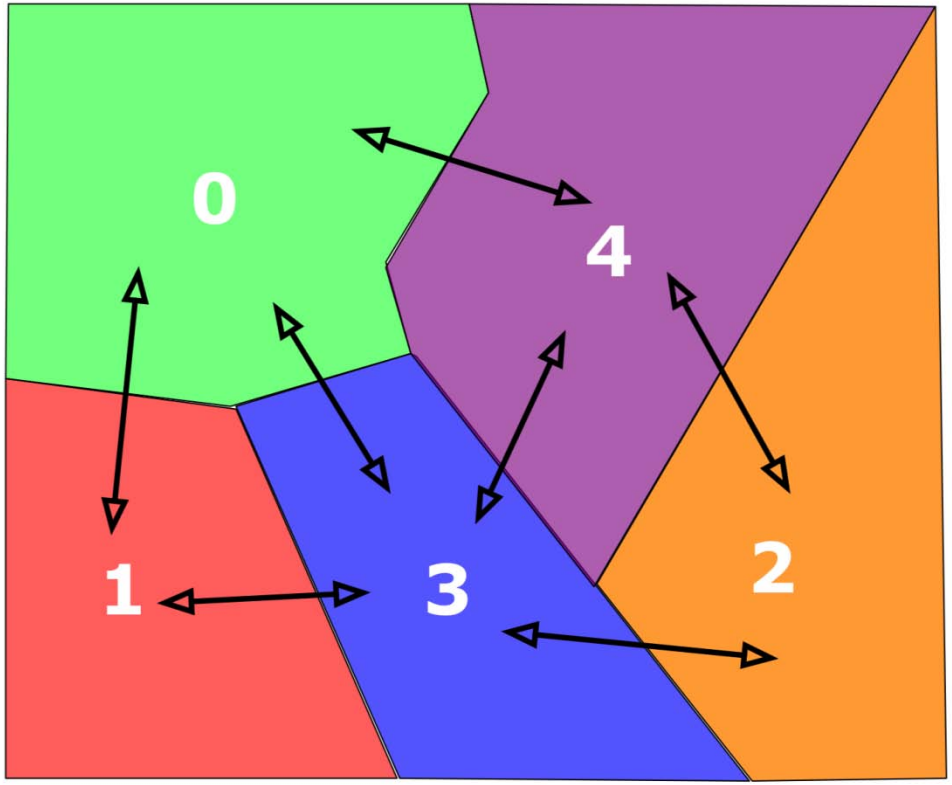


IV- The Linear Solver

- Only the coefficients of the **Core** elements are assembled
- The values of the variable at the shadow elements is updated from the other partitions
- The type of the solver will be discussed later
- As each processor iterates on its set of data, values at the shadow entities have to be updated
- A communication step is therefore needed
- However, this time communication has to be extremely efficient since it will determine the overall efficiency of the solver

IV- The Linear Solver

- The problem with this communication step is processor scheduling



The need for scheduling arises from the fact that all processors are involved in the communication.

For example, while partition 0 is communicating with part. 1, part. 3 will be communicating w/t part. 0, 4 w/t 0,2 w/t 3 and so on…

IV- The Linear Solver

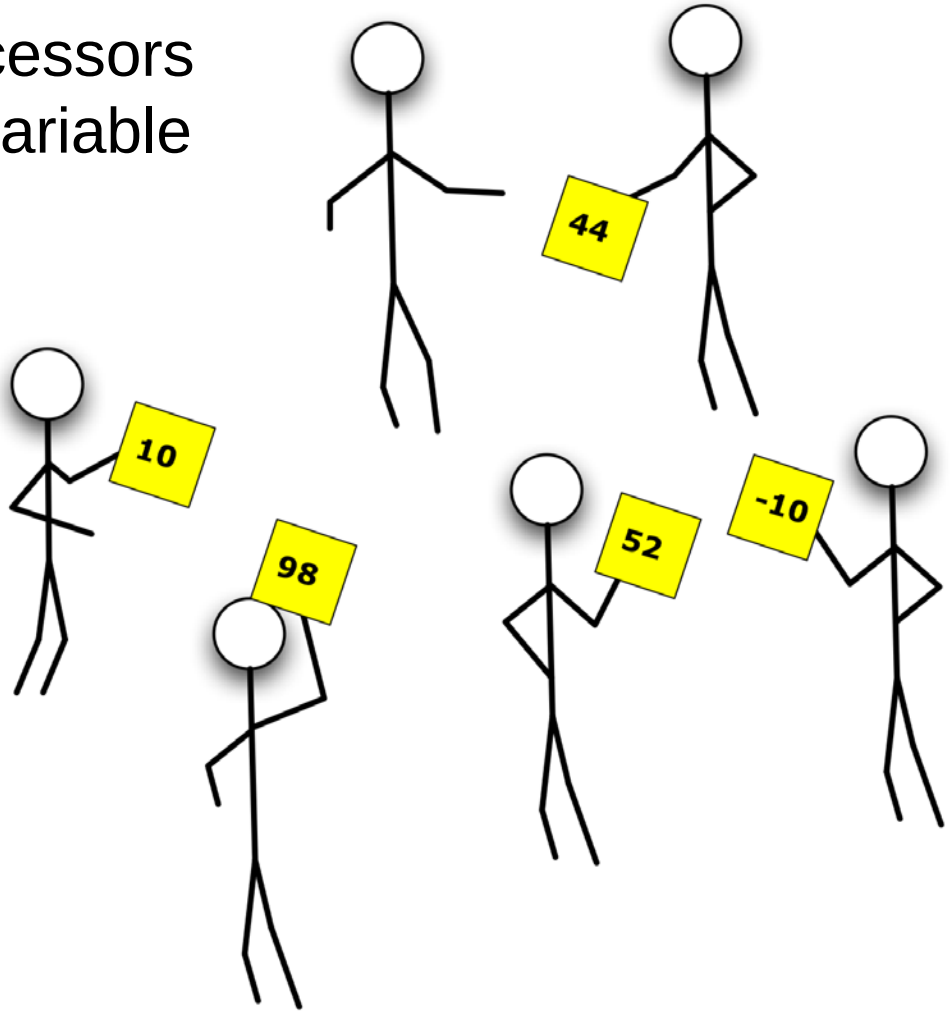
- We have implemented 3 scheduling algorithms:
  - ***Asynchronous***: All partitions post immediate receives from all their neighbors, then again, they post immediate sends to all neighbors
  - ***Semi-Synchronous***: A partition posts an IR from the neighbor with higher index then they post an IS from the neighbor with lower index. In a second loop, they do the opposite, send to higher then receive from lower.
  - ***Pseudo-Synchronous***: Uses a special built-in function that assures no deadlock
  - ***Fully-Synchronous***: A graph theory problem! To be investigated…

IV- The Linear Solver

- Finally, a decision has to be made on whether we have reached convergence or not
- This is done through synchronization of the residuals
- This involves another communication step
- This can be accomplished through:
  - All processors send their residuals to the MASTER
  - The MASTER selects the maximum residual
  - The MASTER then sends this max residual to all processors
- It would cumbersome (stupid) to use P2P communication since the same variable will be distributed to all processors
- A better approach is to use ***COLLECTIVE*** communication

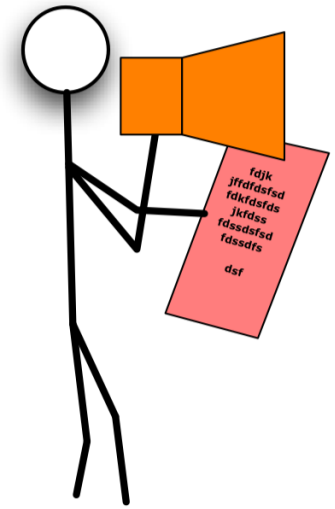
IV- The Linear Solver

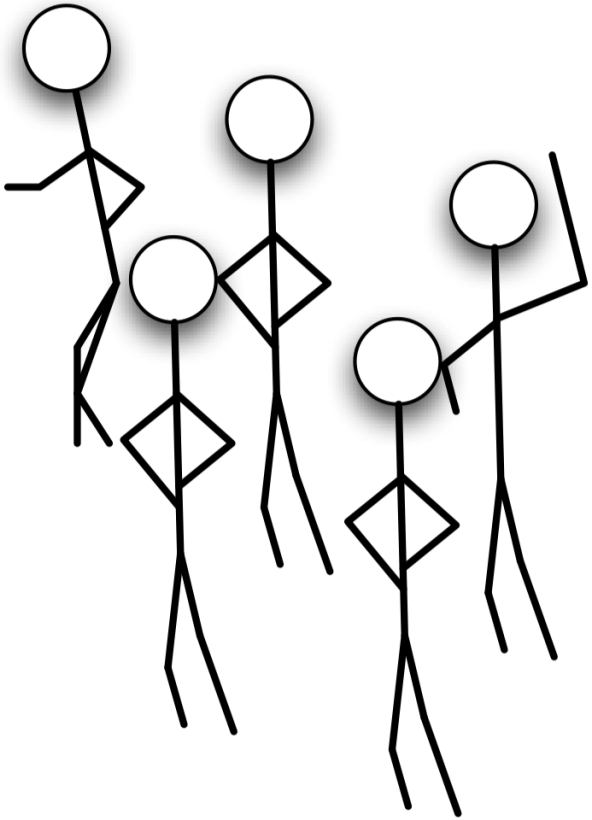
- **REDUCE**: All processors send a specified variable to the MASTER



IV- The Linear Solver

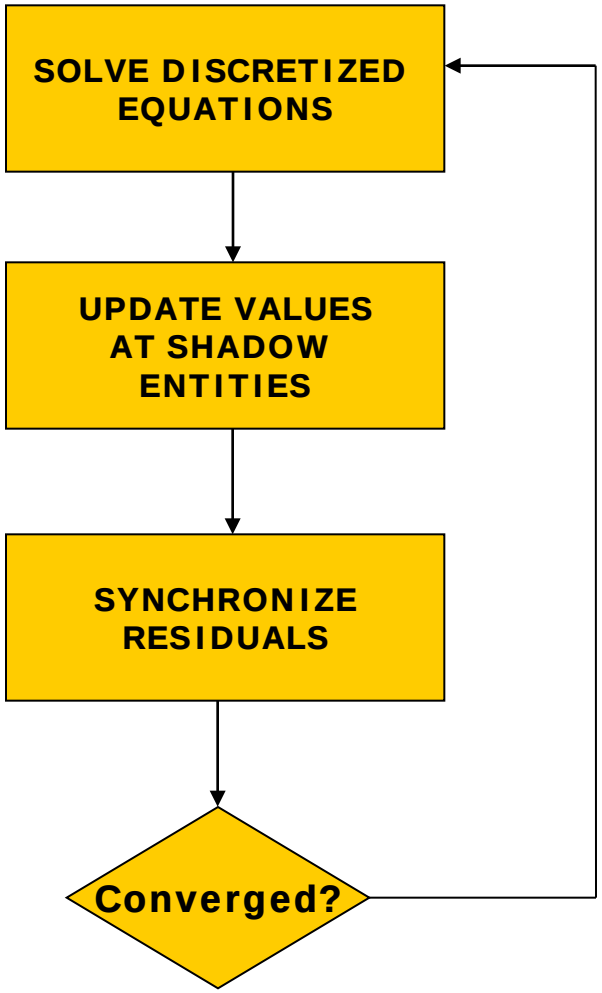
- **BROADCAST**: A root process sends a specified variables to all other processors





IV- The Linear Solver

- To Summarize:



V- The Solution…Finally!

- At this step, all the processors would have obtained a solution (physical?) on their respective partitions
- We need to send this solution to master in order to perform the proper display of data
- This is done in the same way as before
- Nodes send their solution to the master who does the necessary mappings between global and local indices
- This can be done using a collective communication

When the solution is simple, God is answering
(Albert Einstein)

The Outcome:::

- What has been done:
  - Partitioning & Partition Re-Indexing
  - Partition Scheduling Algorithms
  - SOR parallelization
  - ILU Parallelization
  - Parallelization of Diffusion
- Time for a live demo!!!

The End:::

- What has to be done:
  - Parallelization of the Algebraic Multi-Grid solver
  - Implementation of Convection
  - Implementation of pressure correction
  - Implementation of the Energy equation
  - Support for multiphase flows
  - Support for complex models (solidification, Turbulence, Free surface, Star birth…)
- This should require minimal time since the foundation has been setup correctly

.::THE END::.

Thank you for attending