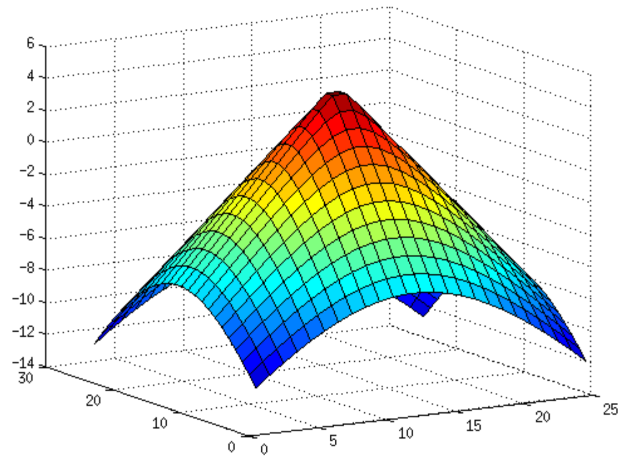
# Parallelization of Level Set Functions Using Message Passing Interface




Institute for Collaborative Biotechnologies


## Presenter

Eric Lee

Contra Costa College

Computer Science

## Lab Mentor

Mohammad Mirzadeh

Computational Applied Science Laboratory (CASL)

## Faculty Advisor

Frederic Gibou

Department of Mechanical Engineering, Computer Science and Mathematics

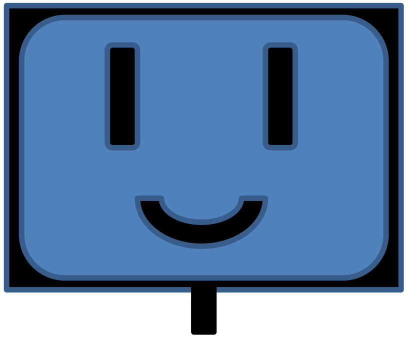
# Topics to be covered

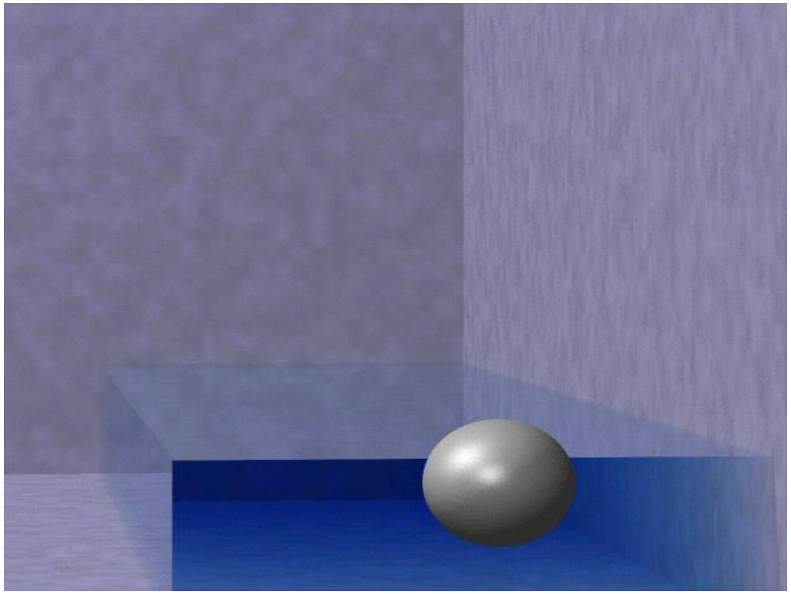
- What are we doing in the Computational Applied Science Laboratory (CASL)?
- What is my project for this summer?
- Serial and Parallel programming.
- Message passing interface.
- Development process.
- Data and progress so far.
- Conclusions and future work.

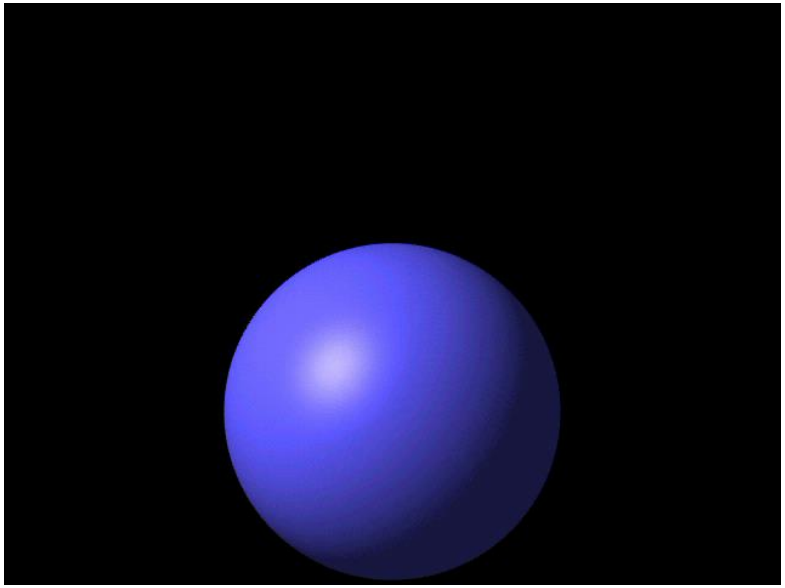
# What are we doing in CASL?

Navier Stokes Equation (General)

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = -\nabla p + \nabla\cdot\mathbf{T} + \mathbf{f},$$

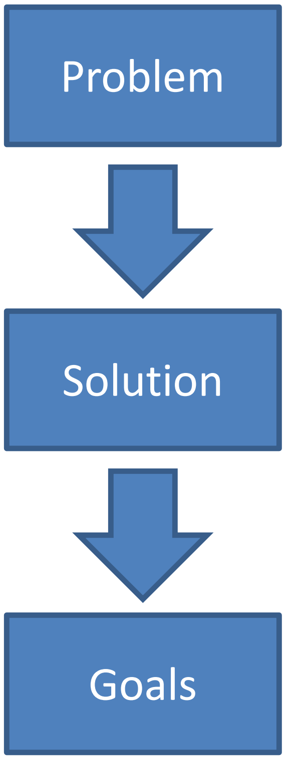






Retrieved from http://www.cs.ucsb.edu/~fgibou/Home.html

# What will I be doing?



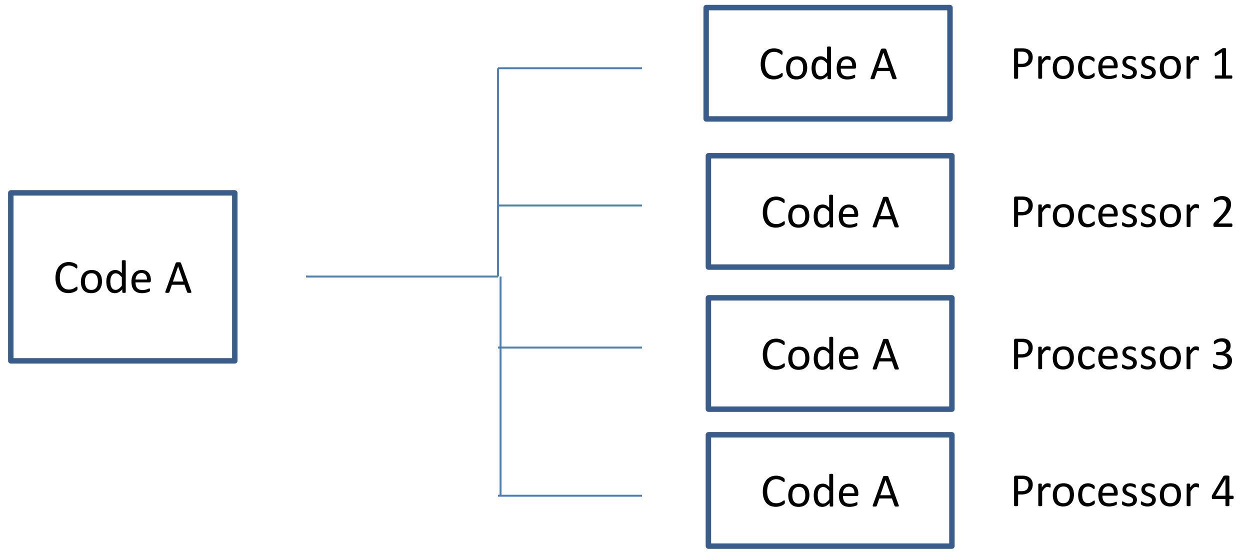

Time!

Parallel programming!

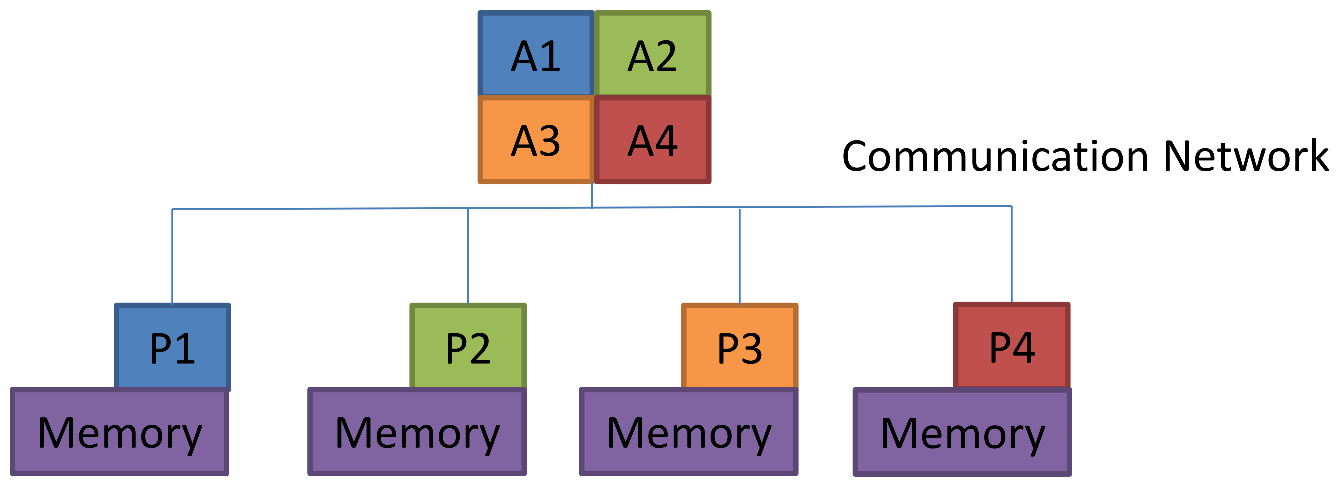
1. Use message passing interface (MPI) to create a serial code that works with one processor.
2. Extend the serial code to work with multiple processors (parallel code)

# Serial and Parallel programming

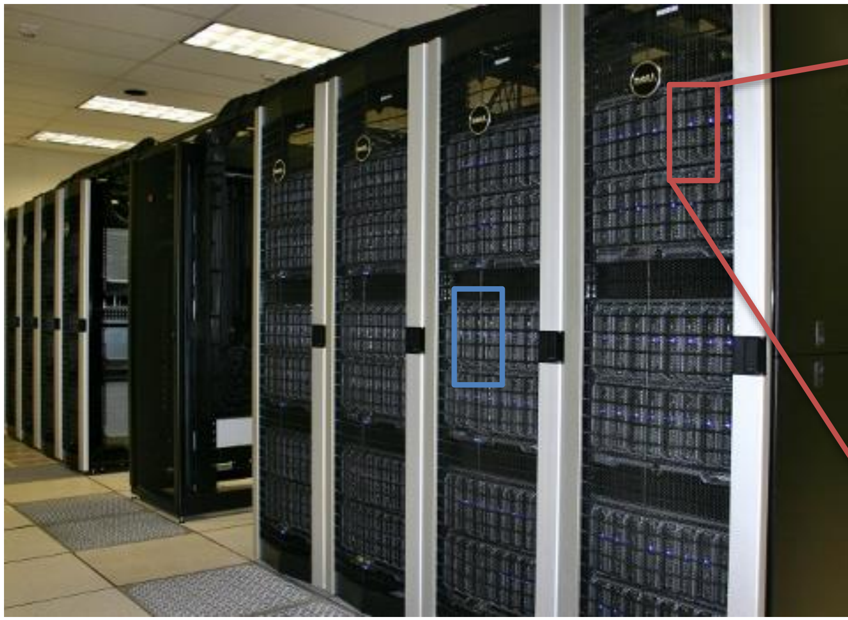
Serial Programming

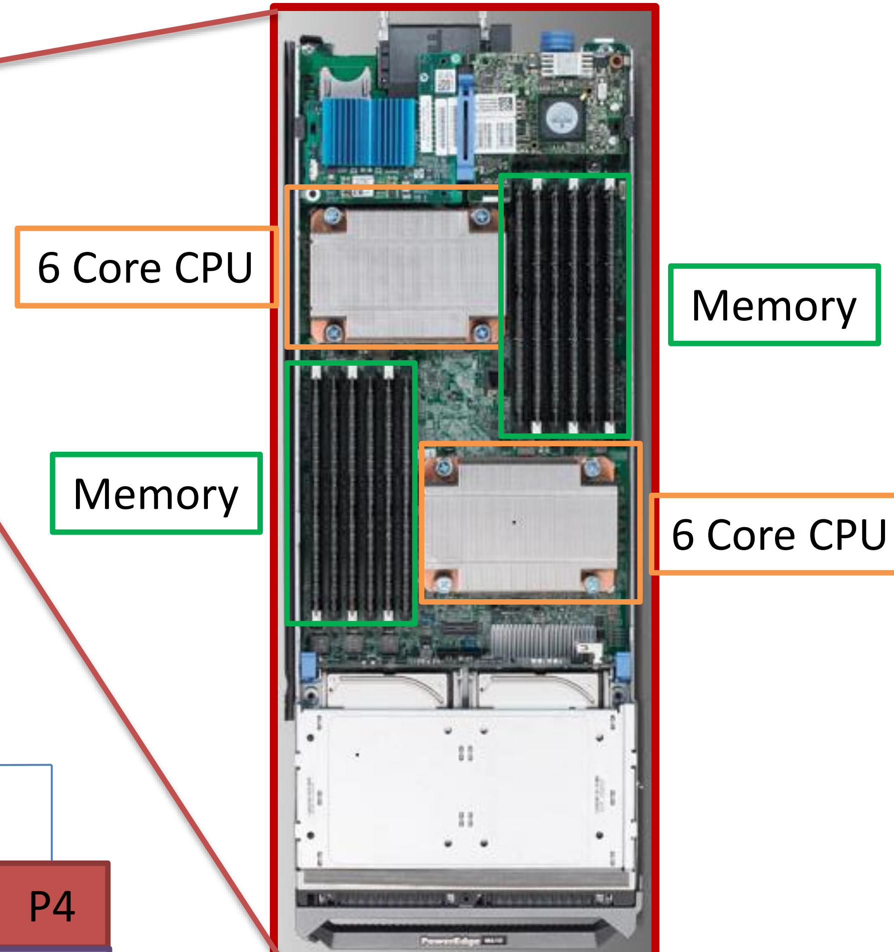


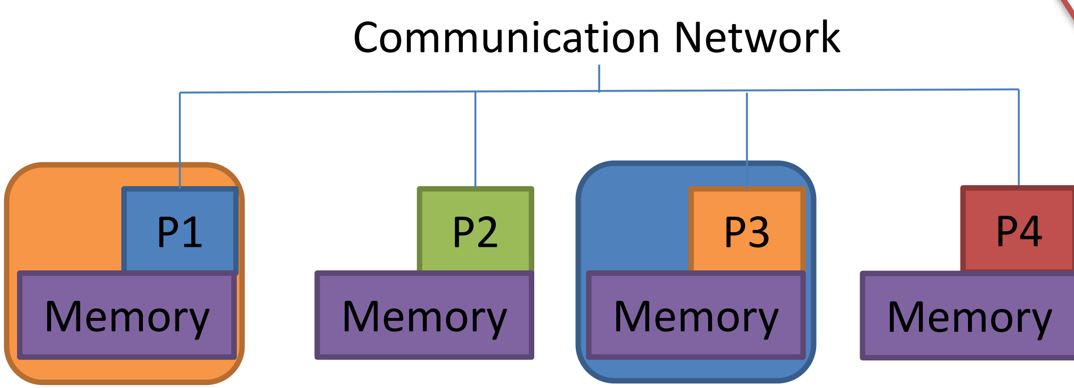

Parallel Programming with MPI



# Texas Advanced Computing Center





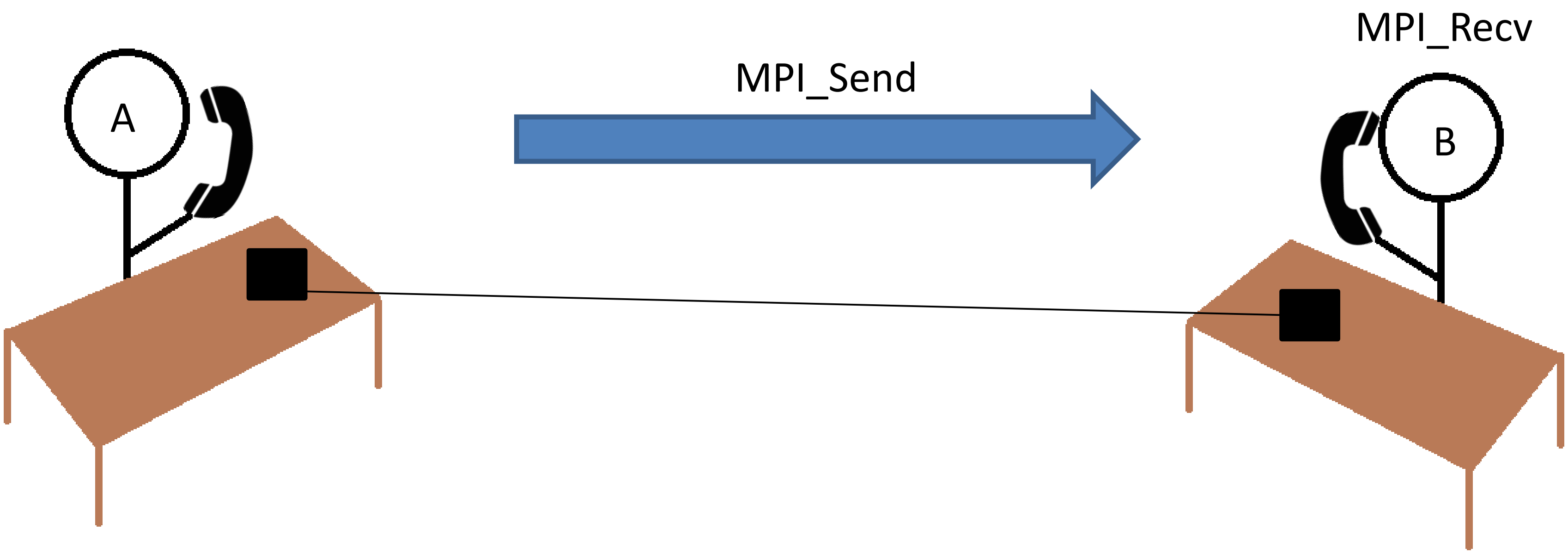




One node

Retrieved from http://www.tacc.utexas.edu/user-services/user-guides/lonestar-user-guide

# What is message passing interface?

- Message passing interface (MPI) is the industry standard for parallelizing code.
- Common MPI commands:
  - MPI_Send
  - MPI_Recv

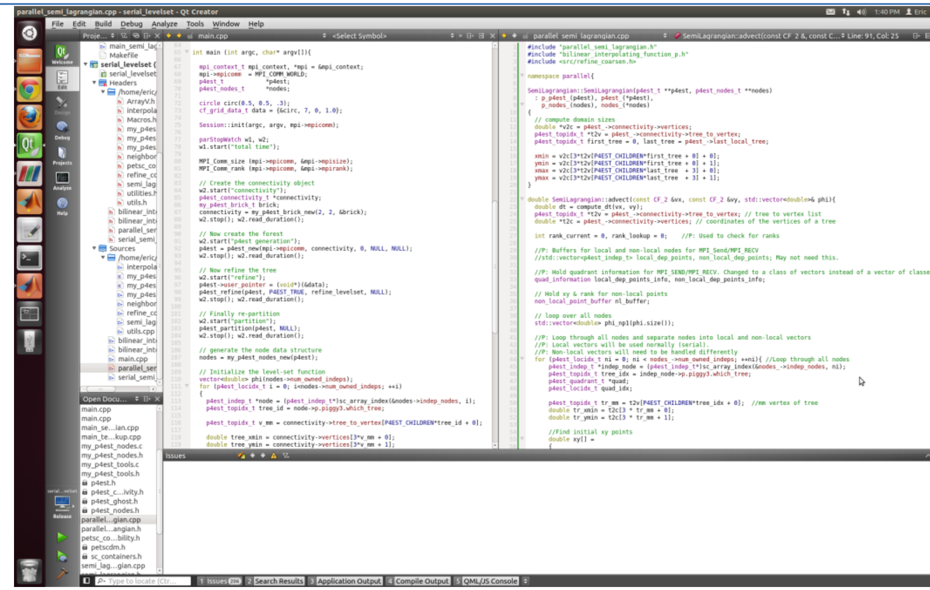


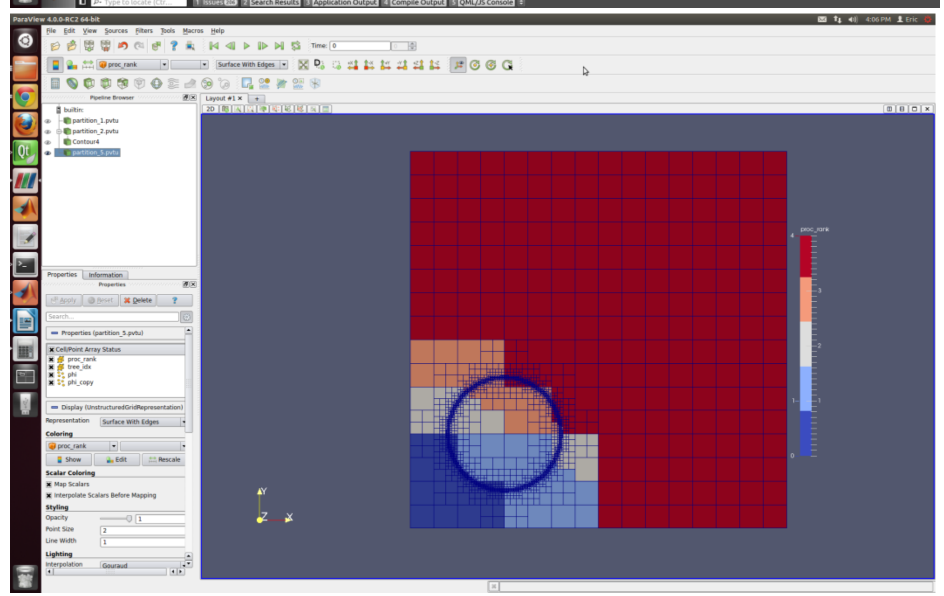
# Development Process

C / C++

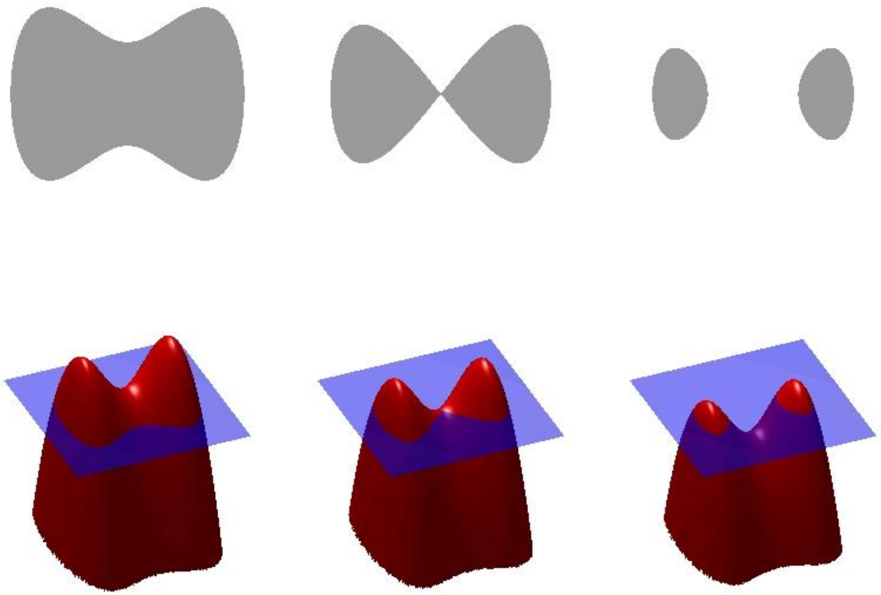
Parallelize code with MPI

ParaView
Parallel Visualization Application

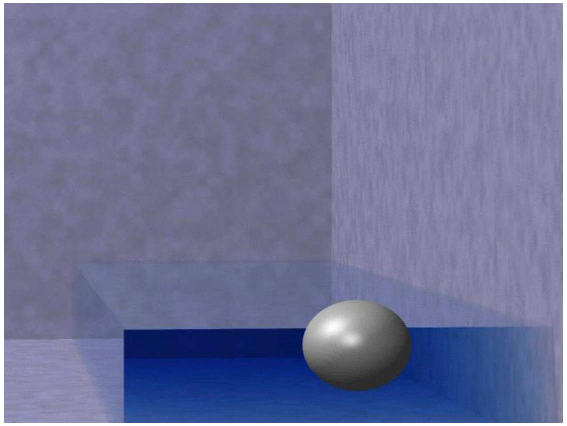


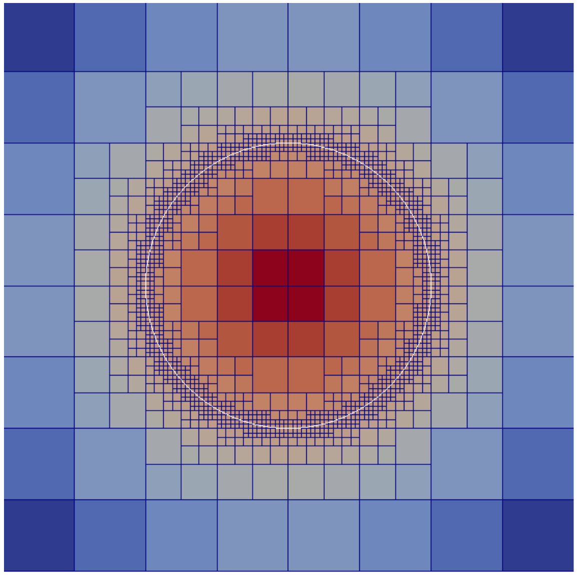


# What are Level Sets?

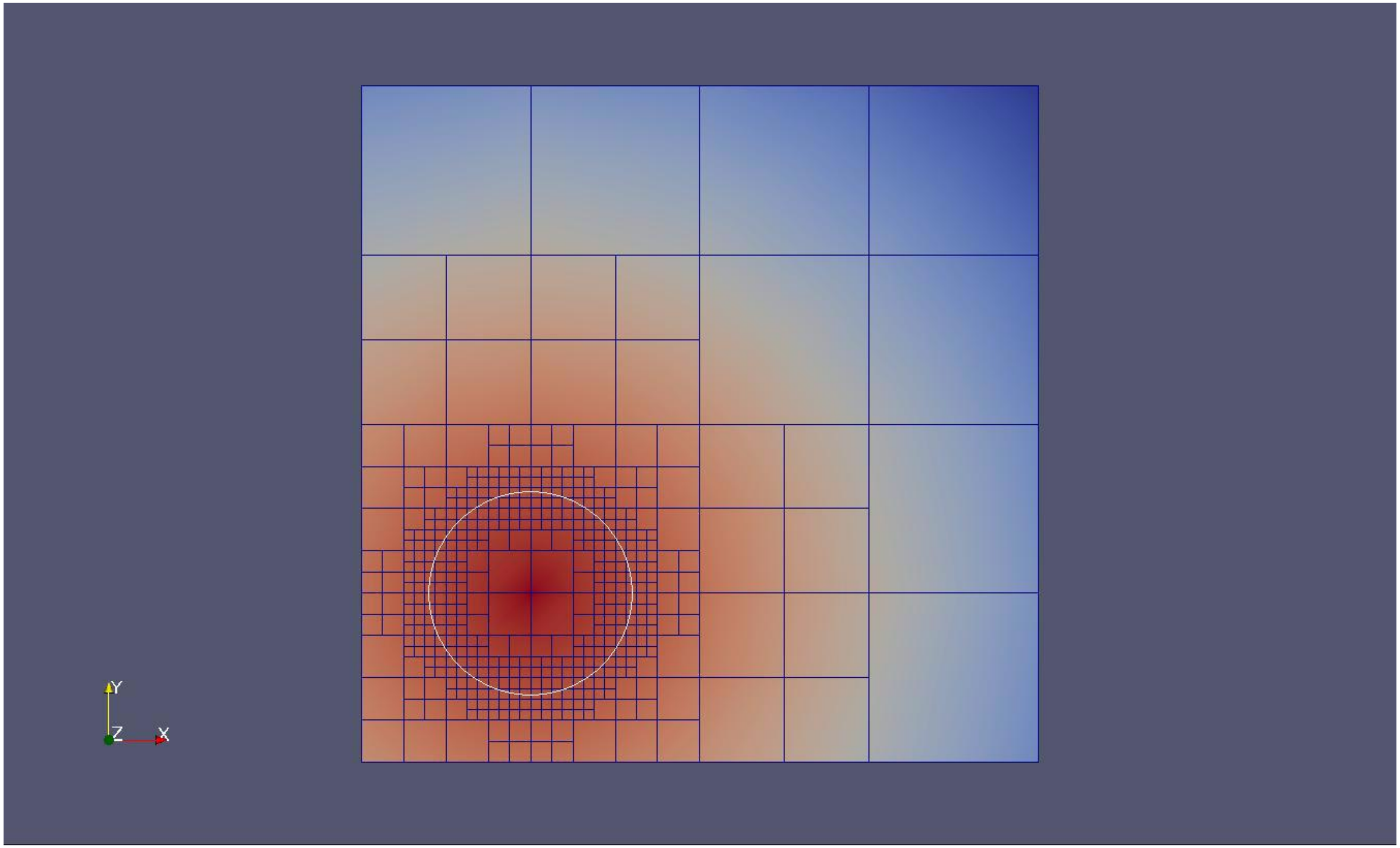


Retrieved from http://en.wikipedia.org/wiki/Level_set_method

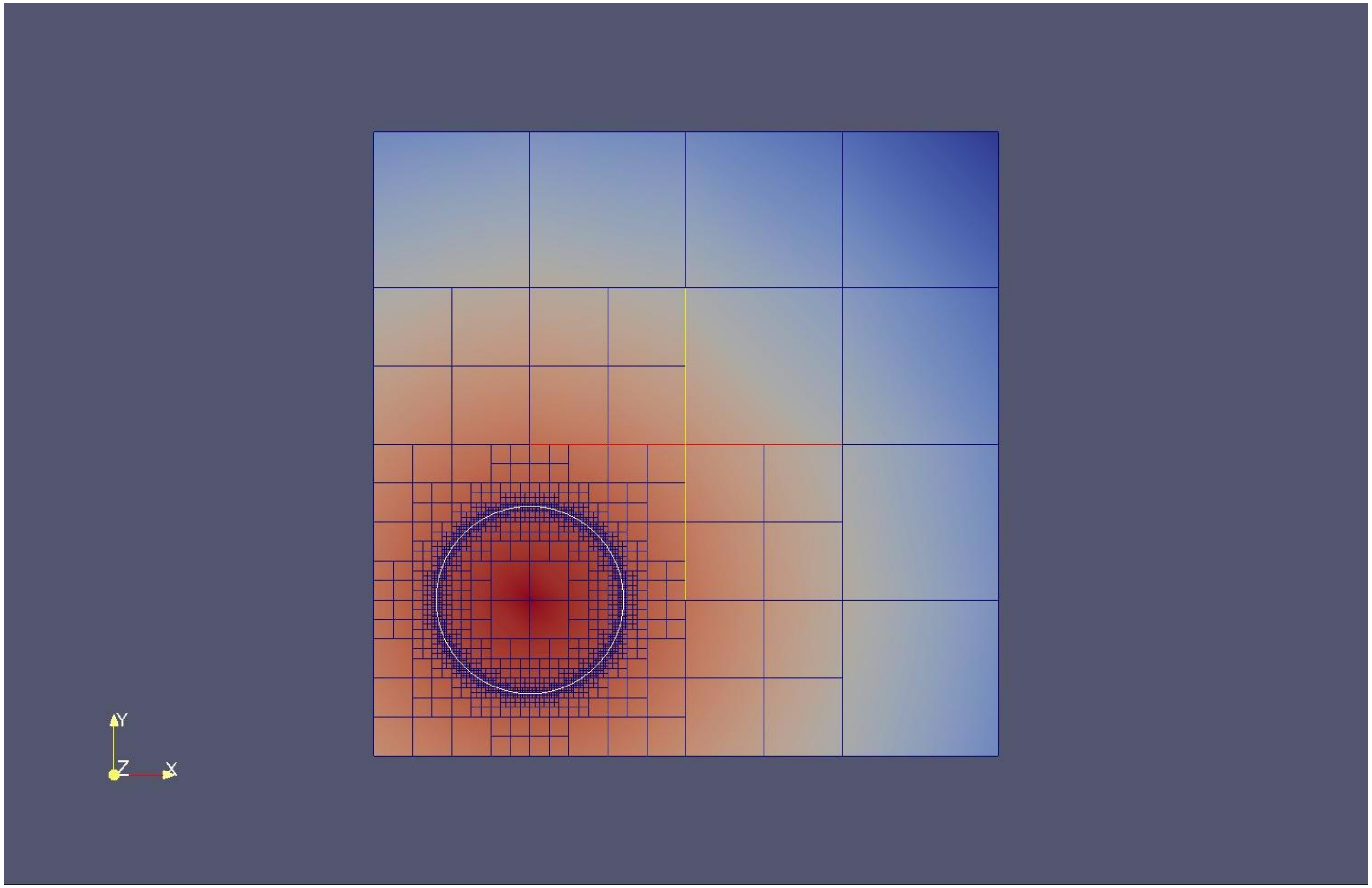




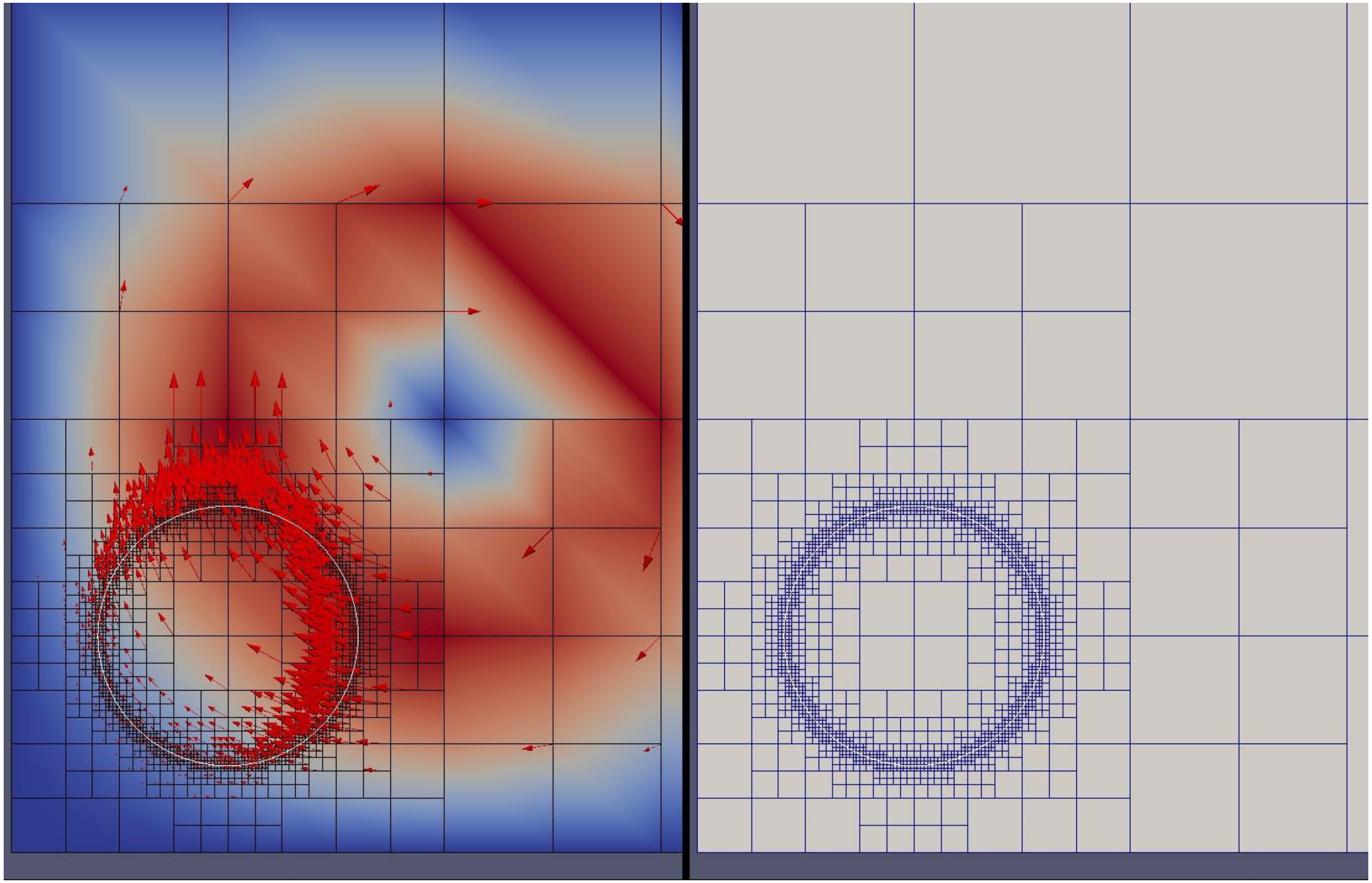
# Level Set Simulation



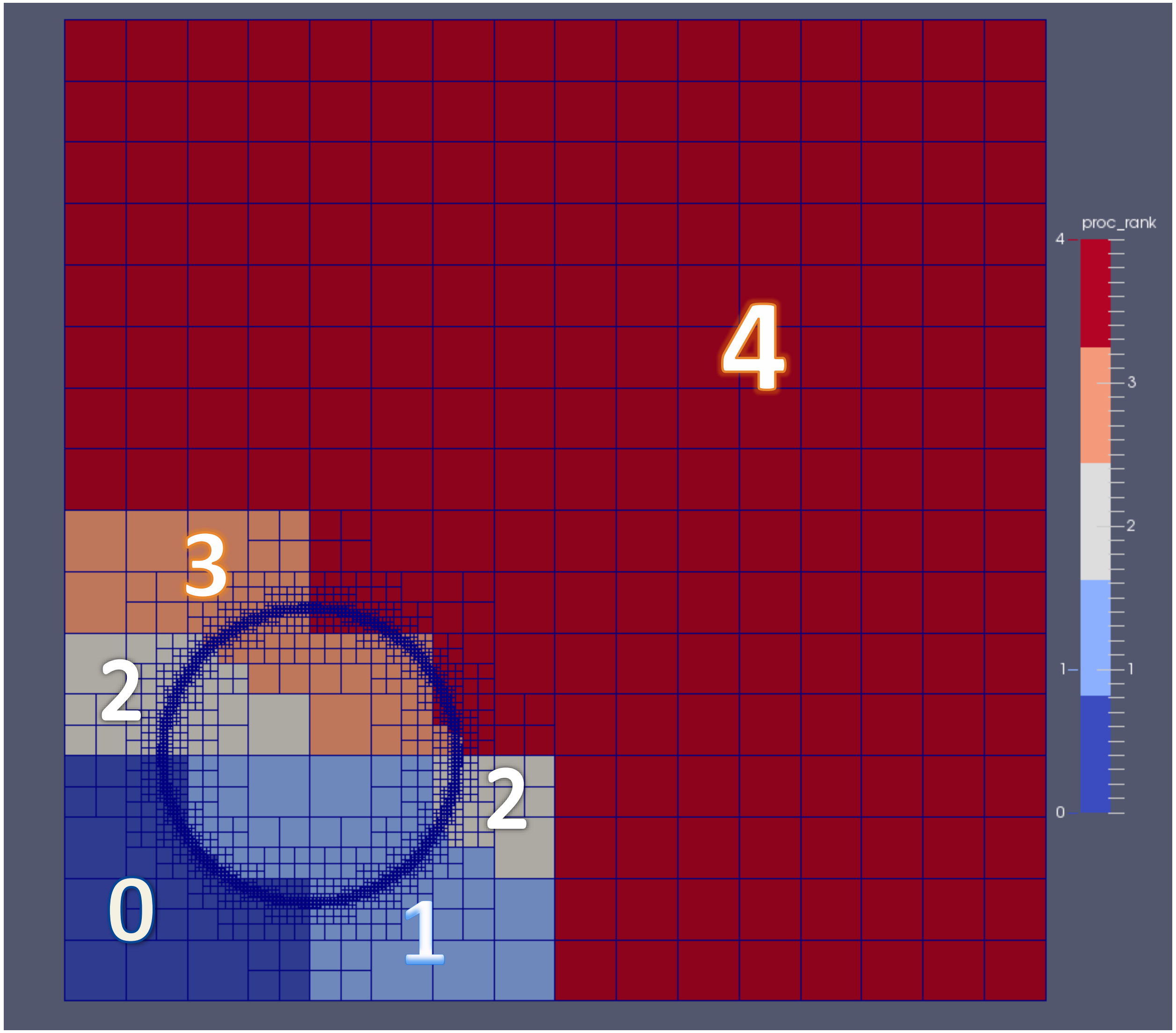
# Buggy simulation



# Level set function with a velocity field



# Partitioning of processor work



# Conclusions and future work

- Serial code is working properly!
- Parallel code works with one processor (essentially a serial code).
- Still need to get parallel code working with multiple processors.
- Compare parallel code to serial code.

Retrieved from http://www.tacc.utexas.edu/user-services/user-guides/lonestar-user-guide

# Acknowledgments

Mentor: Mohammad Mirzadeh

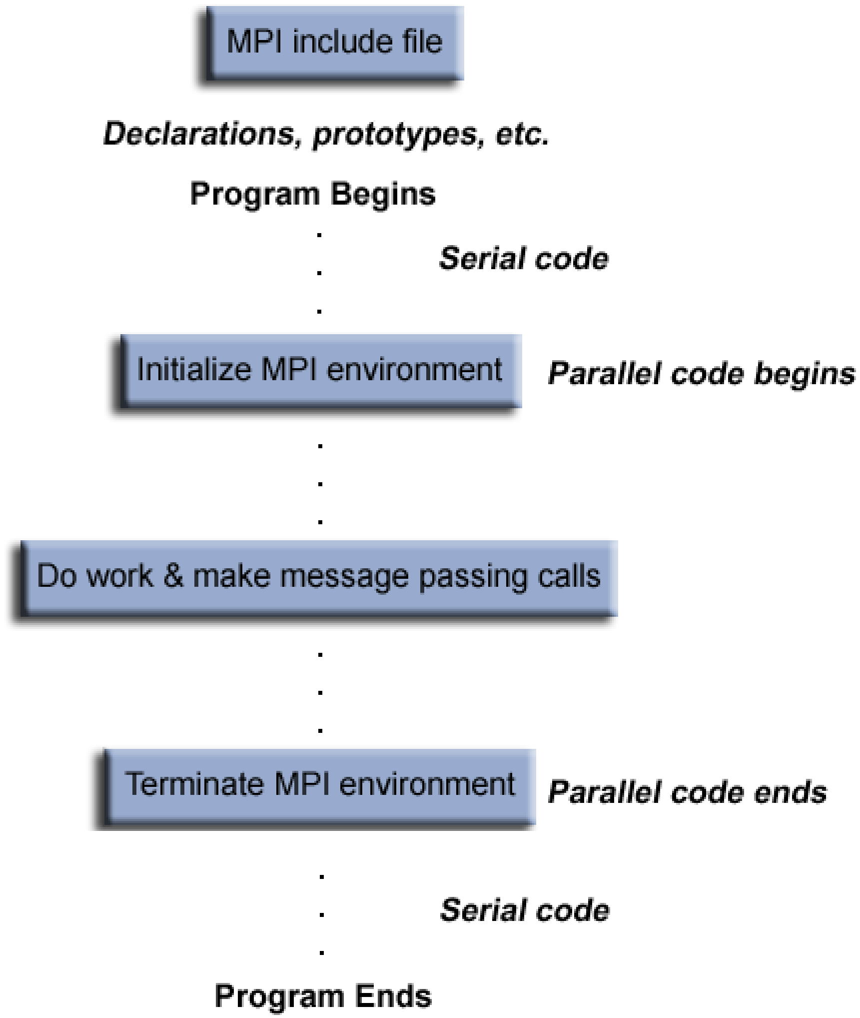
Faculty Advisor: Frederic Gibou

Institute of Collaborative Biotechnologies

National Science Foundation

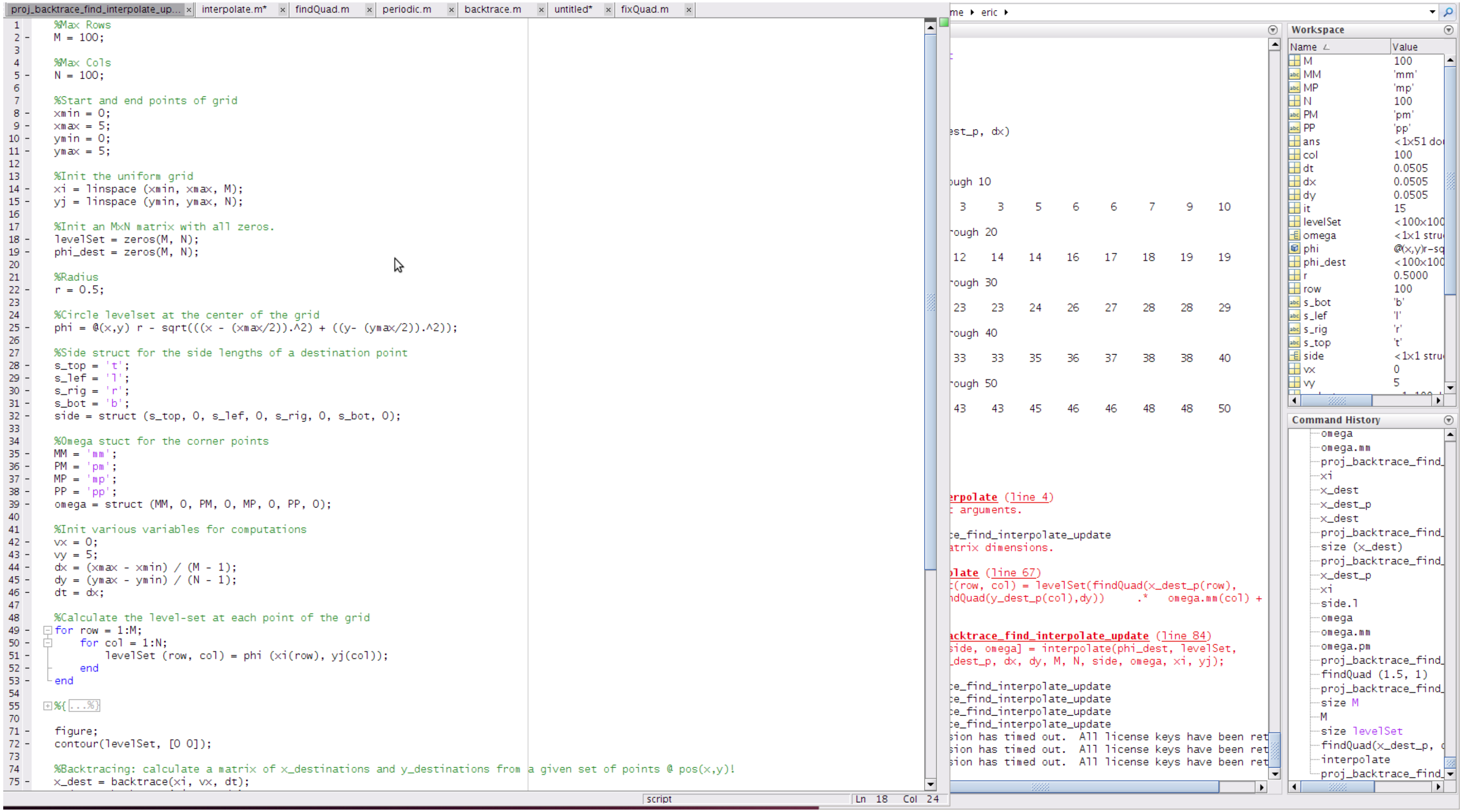
Internships in Nanosystems Science, Engineering, and Technology (INSET)

# MPI Code Development




Retrieved from https://computing.llnl.gov

# Example of MatLab Code



# Example of C++ Code

```cpp

int main (int argc, char* argv[]){

  mpi_context_t mpi_context, *mpi = &mpi_context;
  mpi->mpicomm  = MPI_COMM_WORLD;
  p4est_t            *p4est;
  p4est_nodes_t      *nodes;

  circle circ(0.5, 0.5, .3);
  cf_grid_data_t data = {&circ, 7, 0, 1.0};

  Session::init(argc, argv, mpi->mpicomm);

  parStopWatch w1, w2;
  w1.start("total time");

  MPI_Comm_size (mpi->mpicomm, &mpi->mpisize);
  MPI_Comm_rank (mpi->mpicomm, &mpi->mpirank);

  // Create the connectivity object
  w2.start("connectivity");
  p4est_connectivity_t *connectivity;
  my_p4est_brick_t brick;
  connectivity = my_p4est_brick_new(2, 2, &brick);
  w2.stop(); w2.read_duration();

  // Now create the forest
  w2.start("p4est generation");
  p4est = p4est_new(mpi->mpicomm, connectivity, 0, NULL, NULL);
  w2.stop(); w2.read_duration();

  // Now refine the tree
  w2.start("refine");
  p4est->user_pointer = (void*)(&data);
  p4est_refine(p4est, P4EST_TRUE, refine_levelset, NULL);
  w2.stop(); w2.read_duration();

  // Finally re-partition
  w2.start("partition");
  p4est_partition(p4est, NULL);
  w2.stop(); w2.read_duration();

  // generate the node data structure
  nodes = my_p4est_nodes_new(p4est);

  // Initialize the level-set function
  vector<double> phi(nodes->num_owned_indeps);
  for (p4est_locidx_t i = 0; i<nodes->num_owned_indeps; ++i)
  {
    p4est_indep_t *node = (p4est_indep_t*)sc_array_index(&nodes->indep_nodes, i);
    p4est_topidx_t tree_id = node->p.piggy3.which_tree;

    p4est_topidx_t v_mm = connectivity->tree_to_vertex[P4EST_CHILDREN*tree_id + 0];

    double tree_xmin = connectivity->vertices[3*v_mm + 0];
    double tree_ymin = connectivity->vertices[3*v_mm + 1];
```

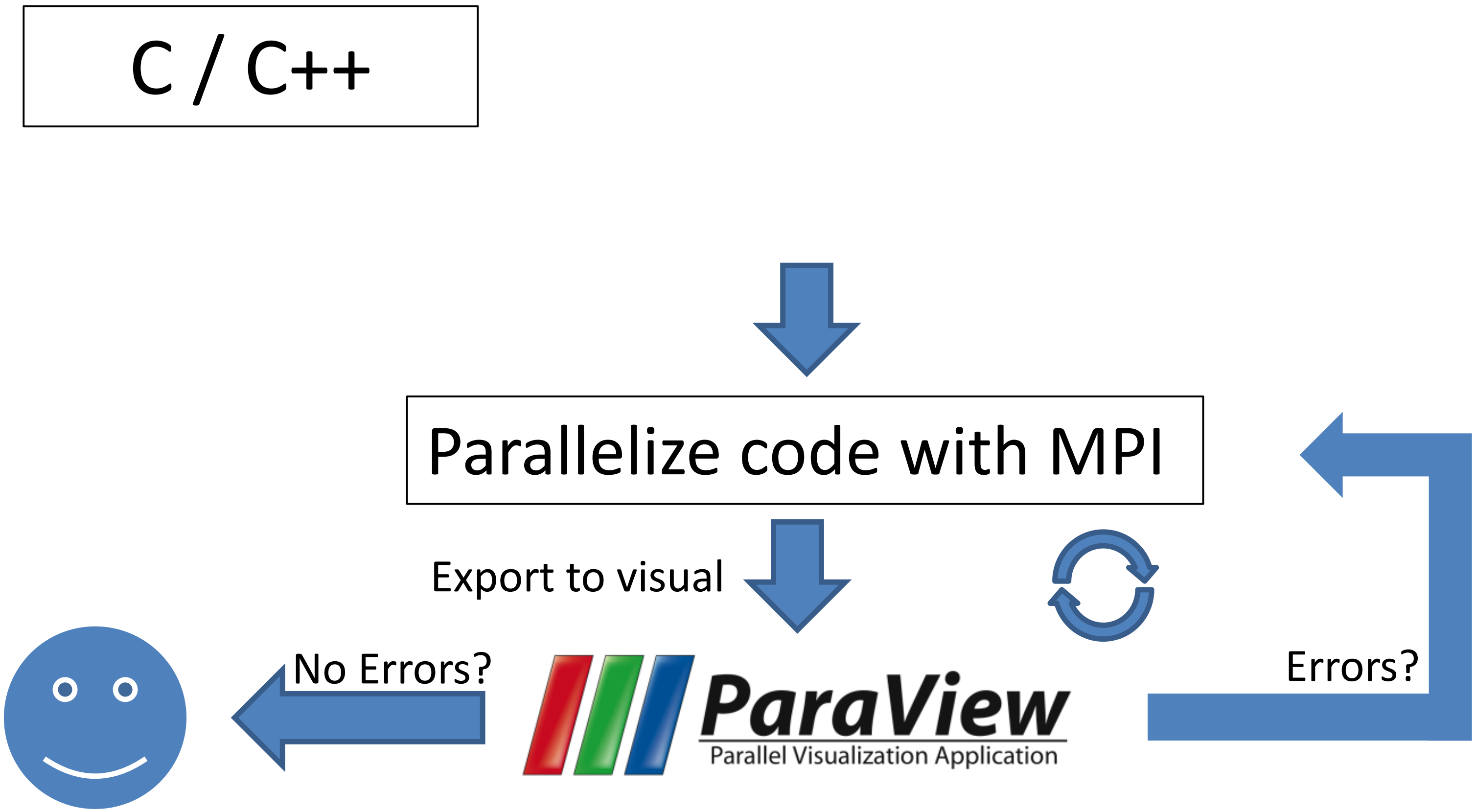
```cpp
#include "parallel_semi_lagrangian.h"
#include "bilinear_interpolating_function_p.h"
#include <src/refine_coarsen.h>

namespace parallel{

SemiLagrangian::SemiLagrangian(p4est_t **p4est, p4est_nodes_t **nodes)
  : p_p4est_(p4est), p4est_(*p4est),
    p_nodes_(nodes), nodes_(*nodes)
{
  // compute domain sizes
  double *v2c = p4est_->connectivity->vertices;
  p4est_topidx_t *t2v = p4est_->connectivity->tree_to_vertex;
  p4est_topidx_t first_tree = 0, last_tree = p4est_->last_local_tree;

  xmin = v2c[3*t2v[P4EST_CHILDREN*first_tree + 0] + 0];
  ymin = v2c[3*t2v[P4EST_CHILDREN*first_tree + 0] + 1];
  xmax = v2c[3*t2v[P4EST_CHILDREN*last_tree  + 3] + 0];
  ymax = v2c[3*t2v[P4EST_CHILDREN*last_tree  + 3] + 1];
}

double SemiLagrangian::advect(const CF_2 &vx, const CF_2 &vy, std::vector<double>& phi){
  double dt = compute_dt(vx, vy);
  p4est_topidx_t *t2v = p4est_->connectivity->tree_to_vertex; // tree to vertex list
  double *t2c = p4est_->connectivity->vertices; // coordinates of the vertices of a tree

  int rank_current = 0, rank_lookup = 0;    //P: Used to check for ranks

  //P: Buffers for local and non-local nodes for MPI_Send/MPI_RECV
  //std::vector<p4est_indep_t> local_dep_points, non_local_dep_points; May not need this.

  //P: Hold quadrant information for MPI_SEND/MPI_RECV. Changed to a class of vectors instead of a vector of classes
  quad_information local_dep_points_info, non_local_dep_points_info;

  // Hold xy & rank for non-local points
  non_local_point_buffer nl_buffer;

  // loop over all nodes
  std::vector<double> phi_np1(phi.size());

  //P: Loop through all nodes and separate nodes into local and non-local vectors
  //P: Local vectors will be used normally (serial).
  //P: Non-local vectors will need to be handled differently
  for (p4est_locidx_t ni = 0; ni < nodes_->num_owned_indeps; ++ni){ //Loop through all nodes
      p4est_indep_t *indep_node = (p4est_indep_t*)sc_array_index(&nodes_->indep_nodes, ni);
      p4est_topidx_t tree_idx = indep_node->p.piggy3.which_tree;
      p4est_quadrant_t *quad;
      p4est_locidx_t quad_idx;

      p4est_topidx_t tr_mm = t2v[P4EST_CHILDREN*tree_idx + 0];  //mm vertex of tree
      double tr_xmin = t2c[3 * tr_mm + 0];
      double tr_ymin = t2c[3 * tr_mm + 1];

      //Find initial xy points
      double xy[] =
      {
```

# What are we doing in CASL?

- We solve partial differential equations (PDEs) and use their solutions to simulate various physical phenomena!
- What are PDEs?
  - Mathematical equations
- What can we describe using PDEs?
  - Weather
  - Electricity and Magnetism
  - Fluid flow
- Why is this important?
  - Saves money!
    - Allows us to run simulations without actually building anything.

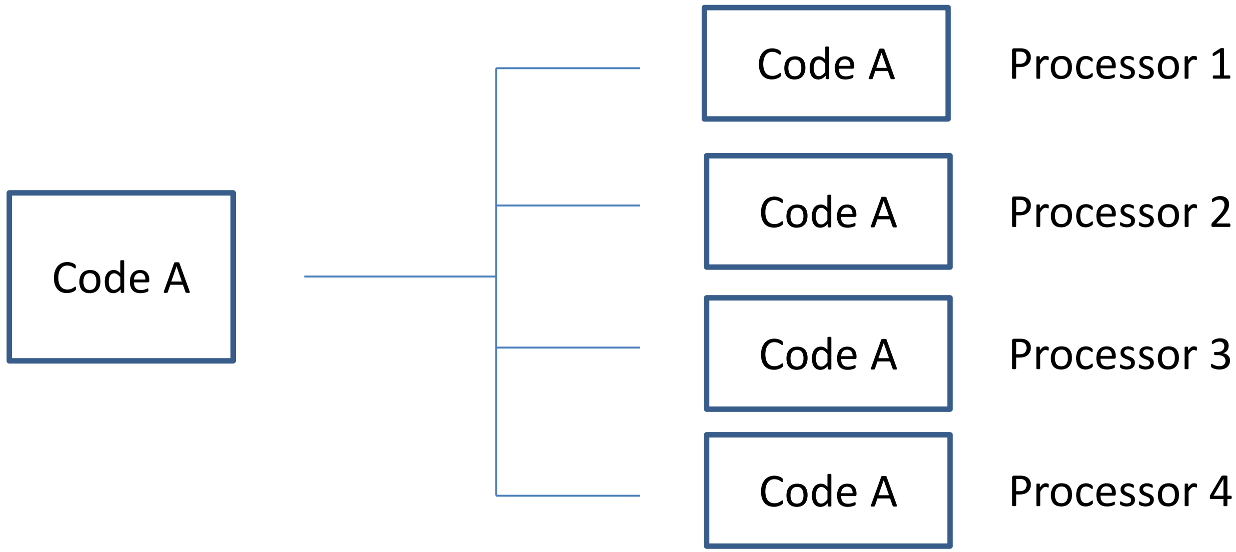
# Development Process

Prepare Serial Code

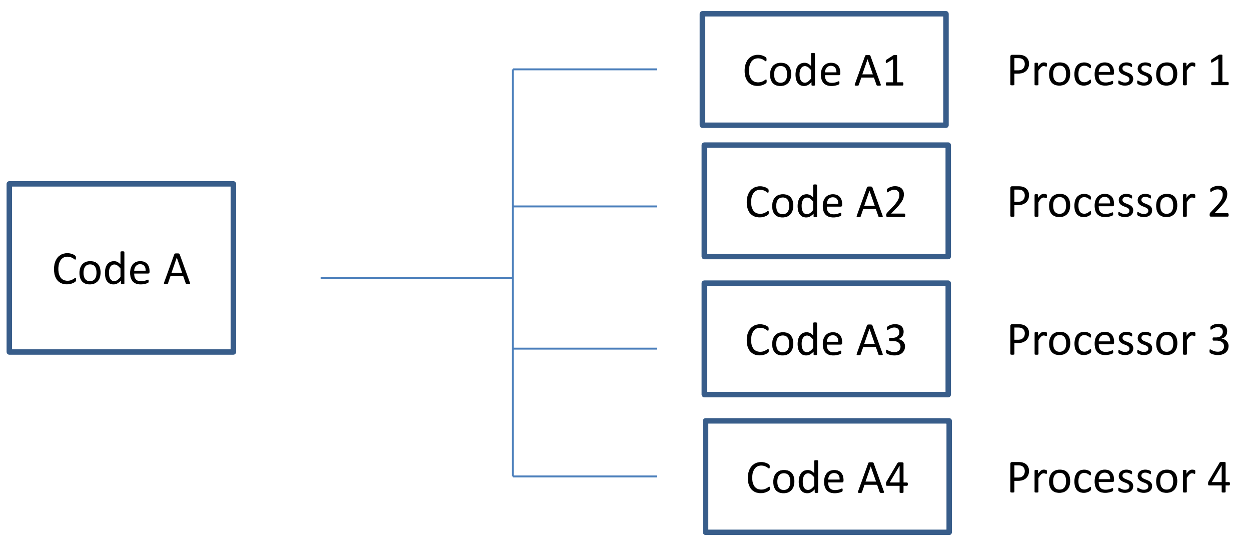


# What is the difference between Serial and Parallel programming?

What is Serial Programming?

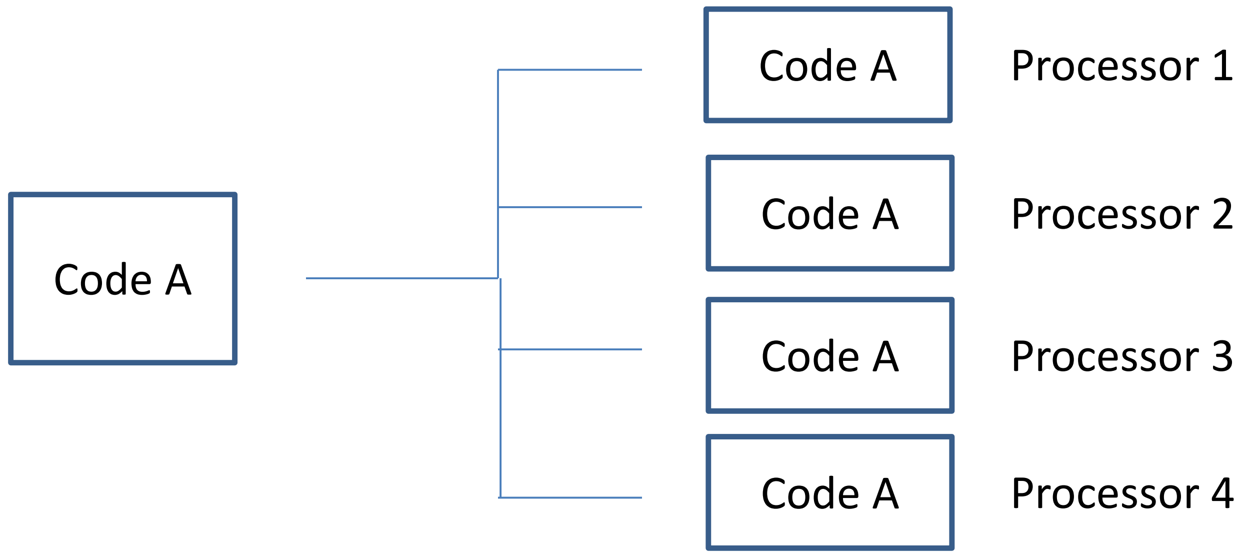



What is Parallel Programming?

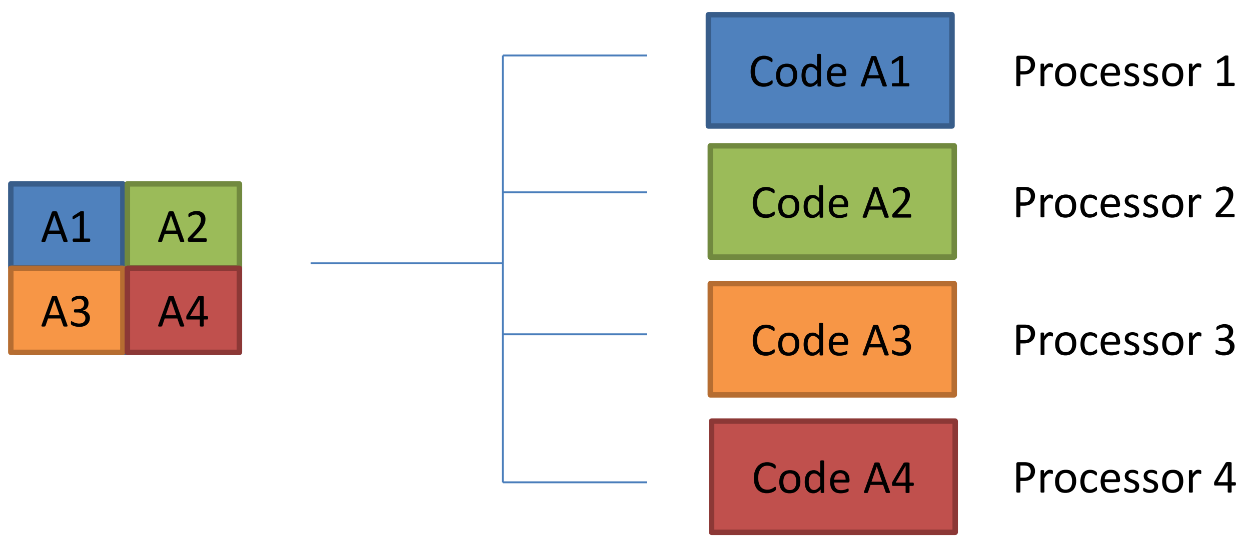


# Parallel programming

What is Serial Programming?




What is Parallel Programming?



# Filler

- Filler