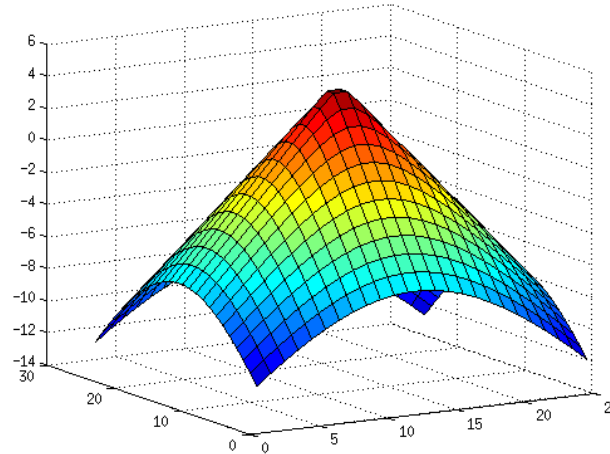


Parallelization of Level Set Functions Using Message Passing Interface



Institute for Collaborative
Biotechnologies

Presenter

Eric Lee
Contra Costa College
Computer Science



Lab Mentor

Mohammad Mirzadeh

Computational
Applied
Science
Laboratory
(CASL)

Faculty Advisor

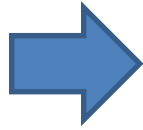
Frederic Gibou

Department of
Mechanical Engineering,
Computer Science and
Mathematics

Topics to be covered

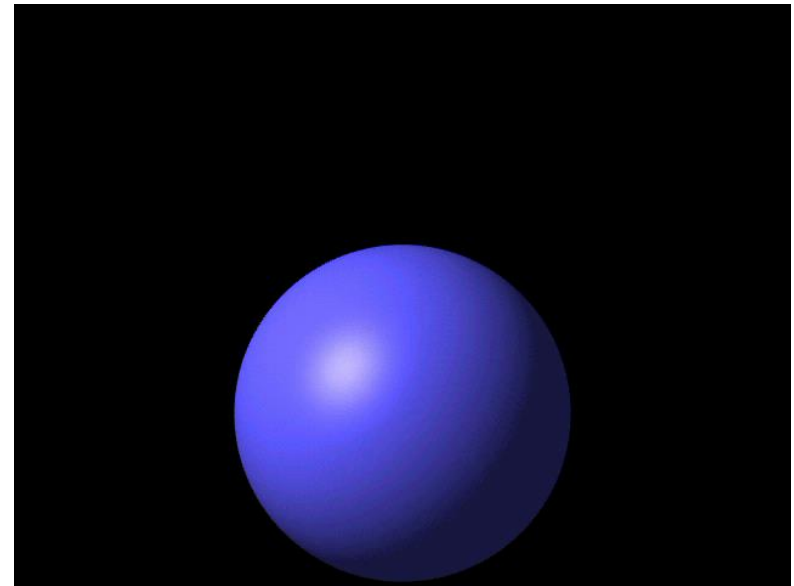
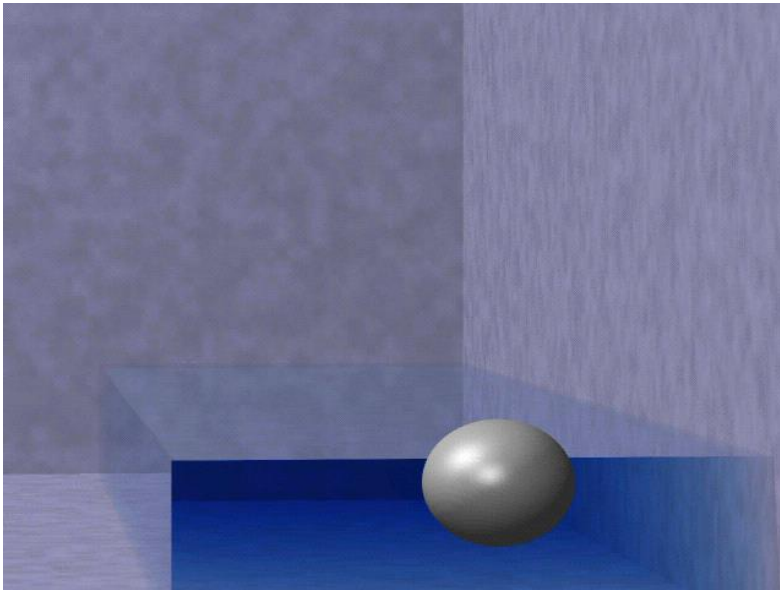
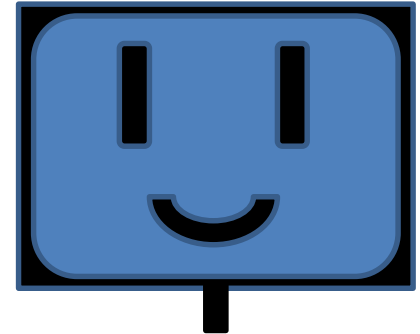
- What are we doing in the Computational Applied Science Laboratory (CASL)?
- What is my project for this summer?
- Serial and Parallel programming.
- Message passing interface.
- Development process.
- Data and progress so far.
- Conclusions and future work.

What are we doing in CASL?

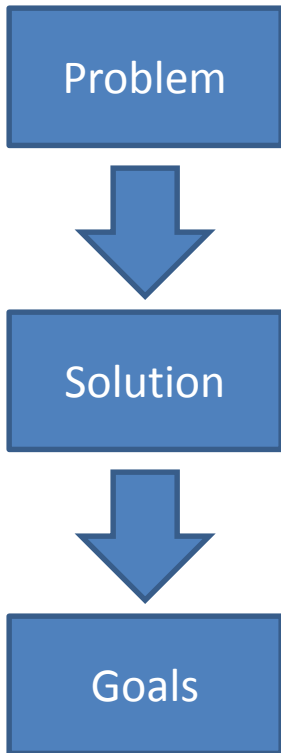


Navier Stokes Equation (General)

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



What will I be doing?

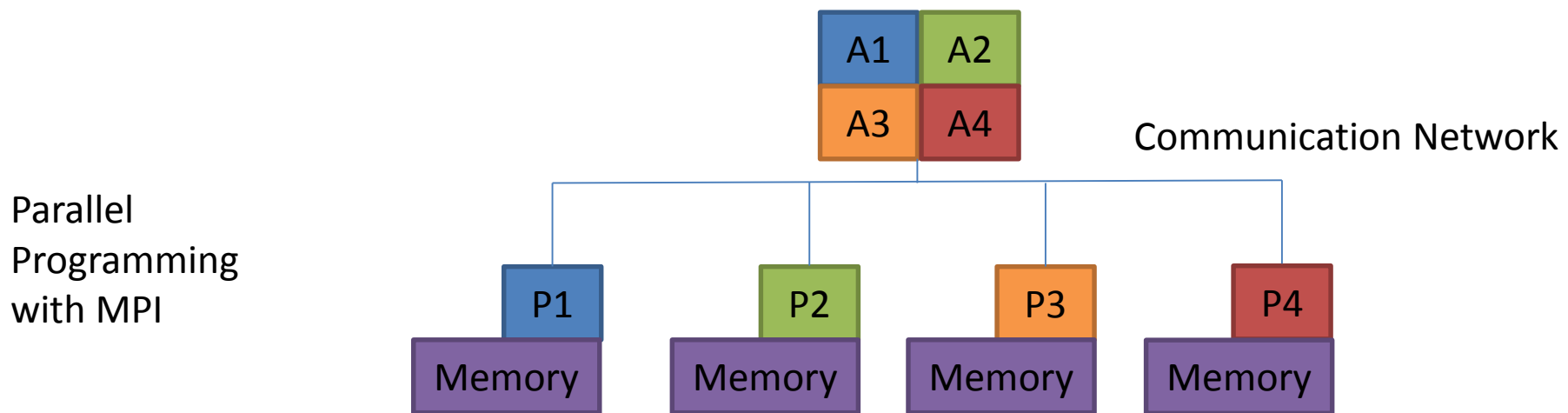
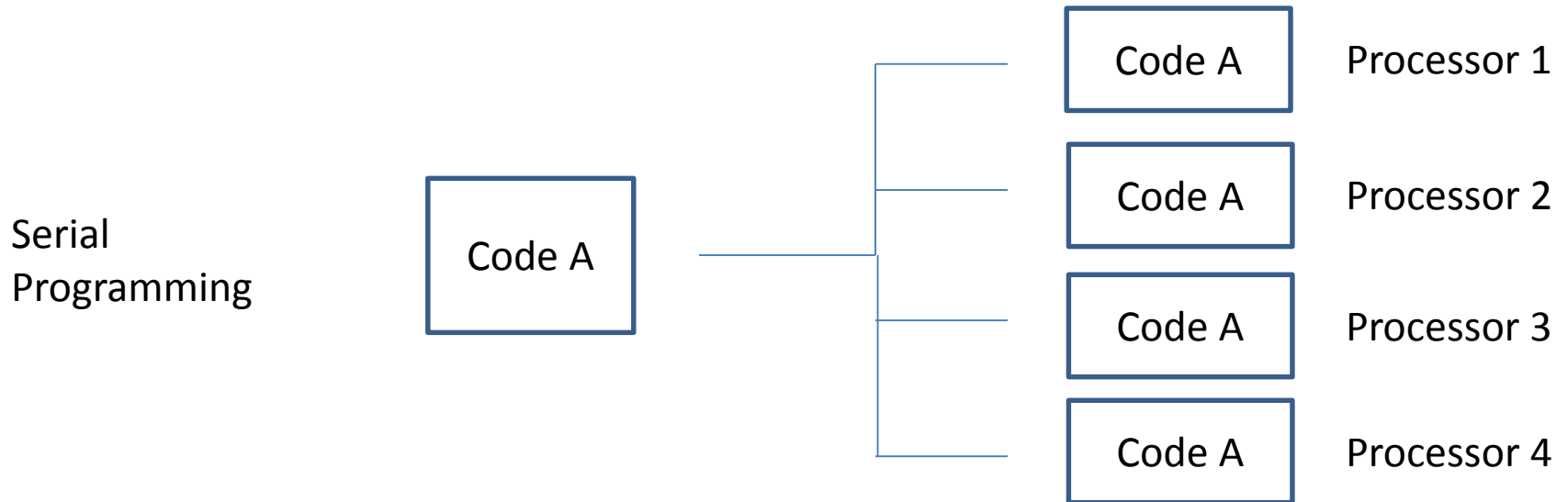


Time!

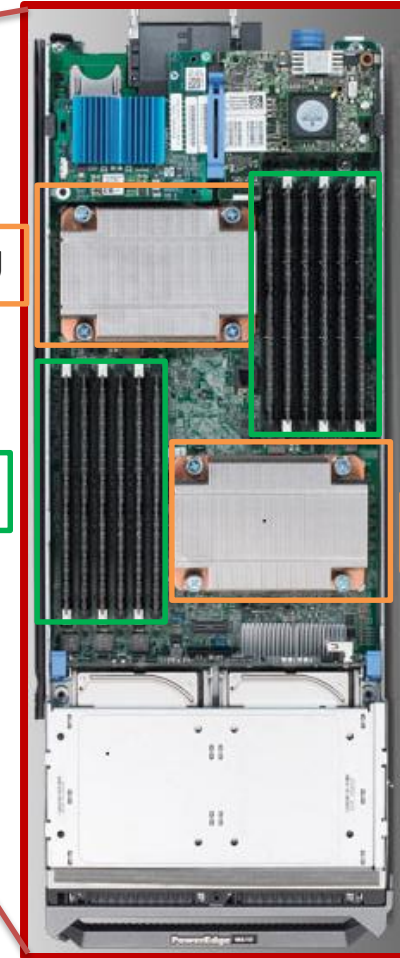
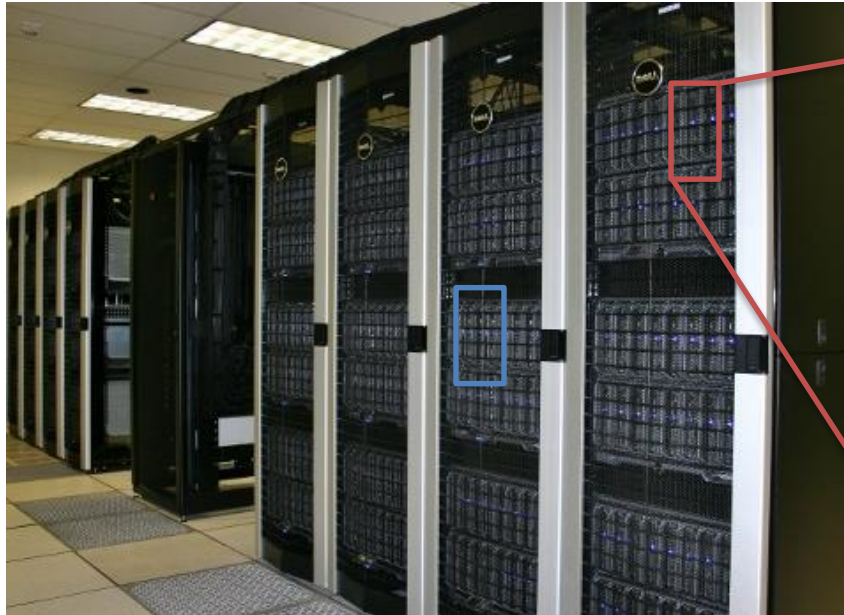
Parallel programming!

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



Texas Advanced Computing Center



6 Core CPU

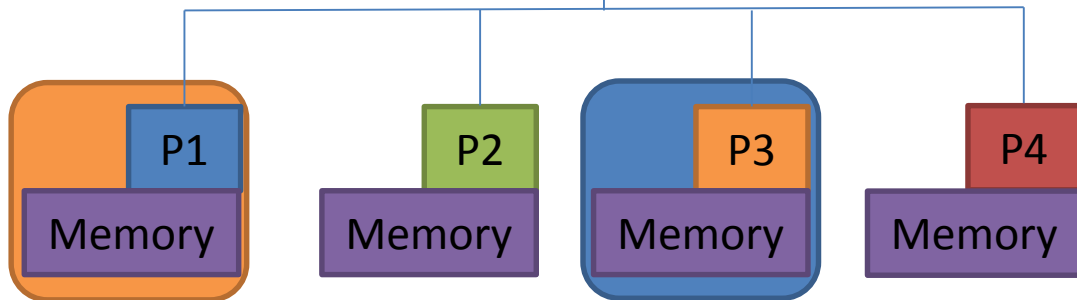
Memory

Memory

6 Core CPU

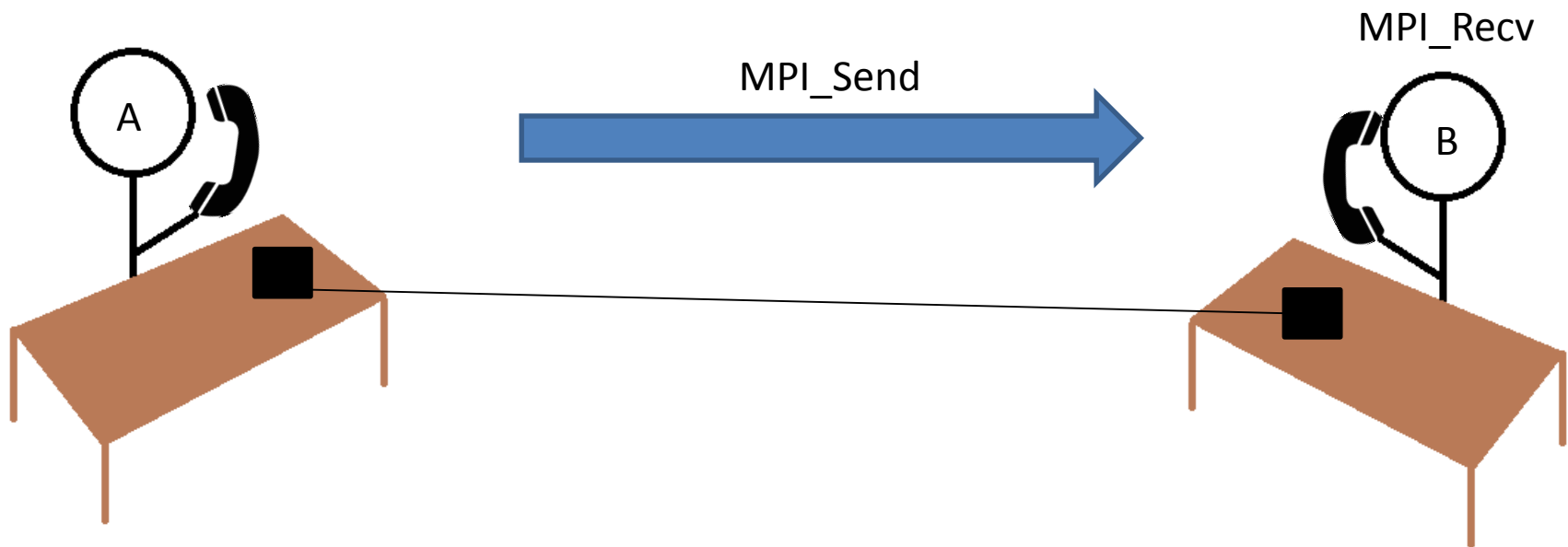
One node

Communication Network



What is message passing interface?

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

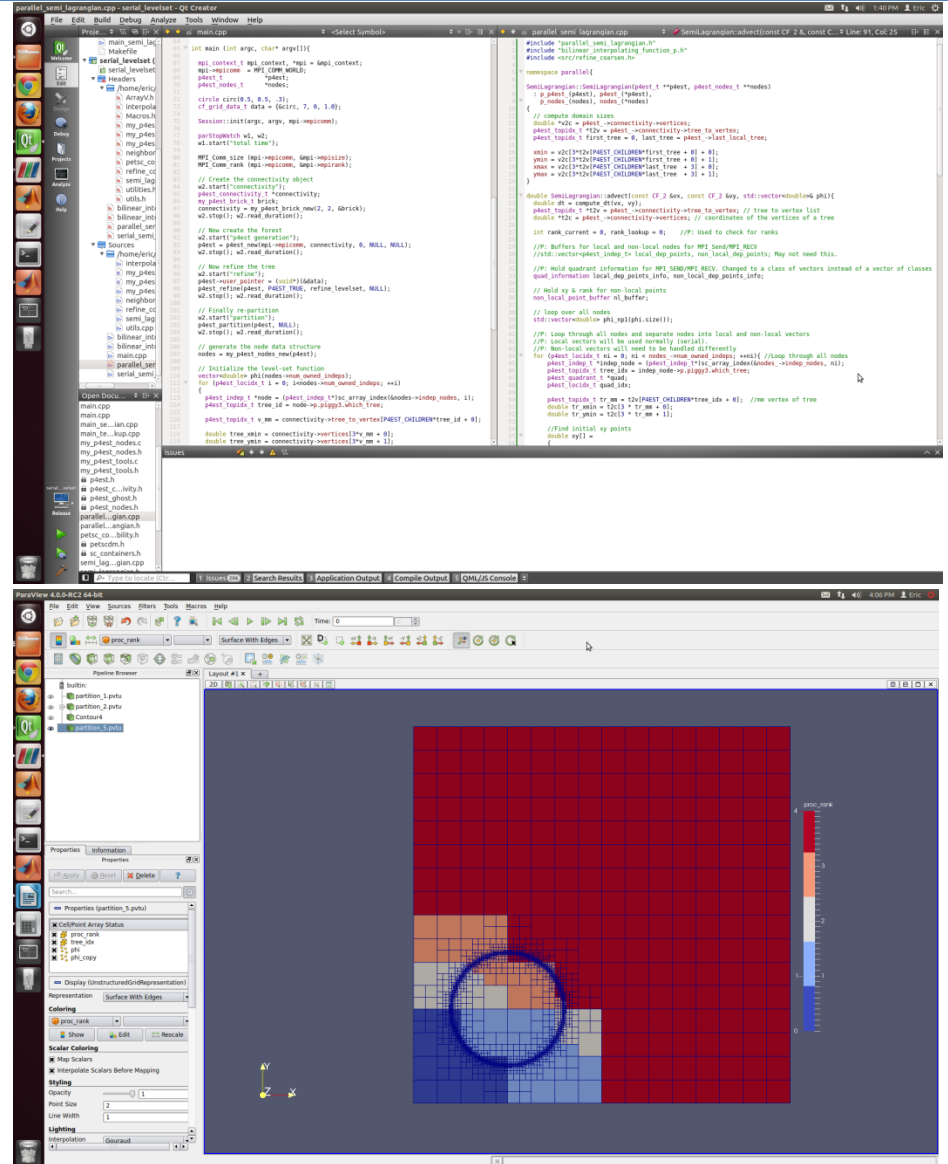


Development Process

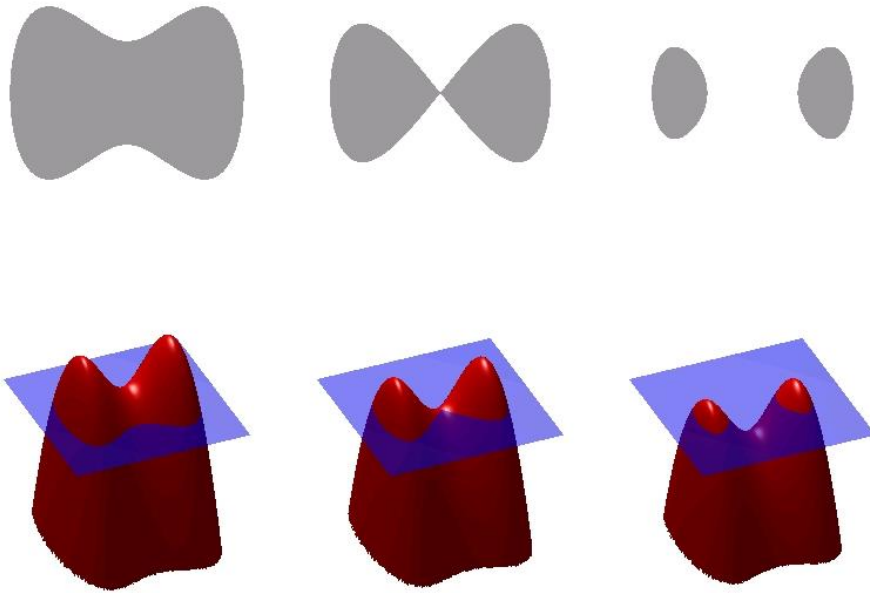
C / C++



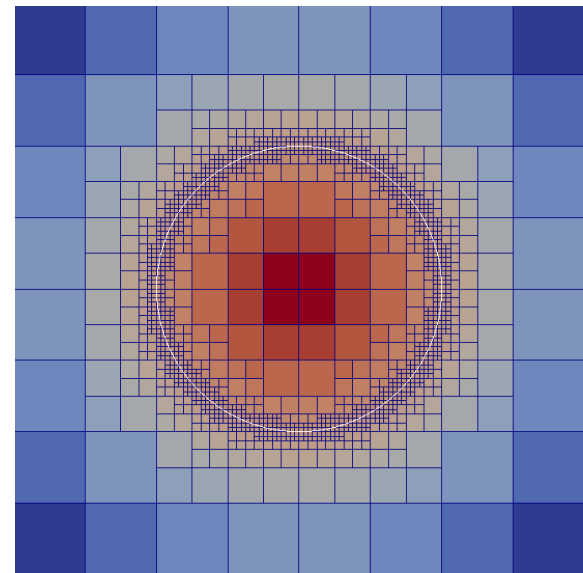
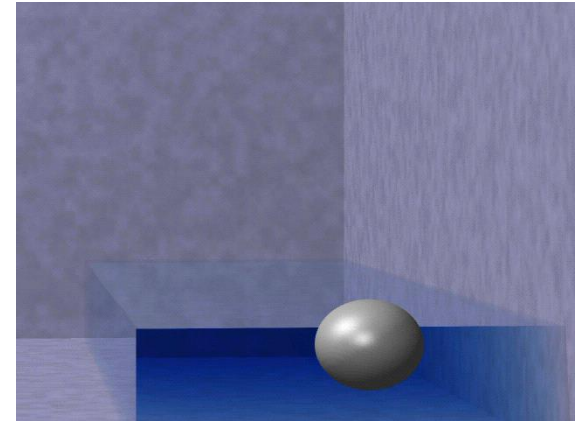
Parallelize code with MPI



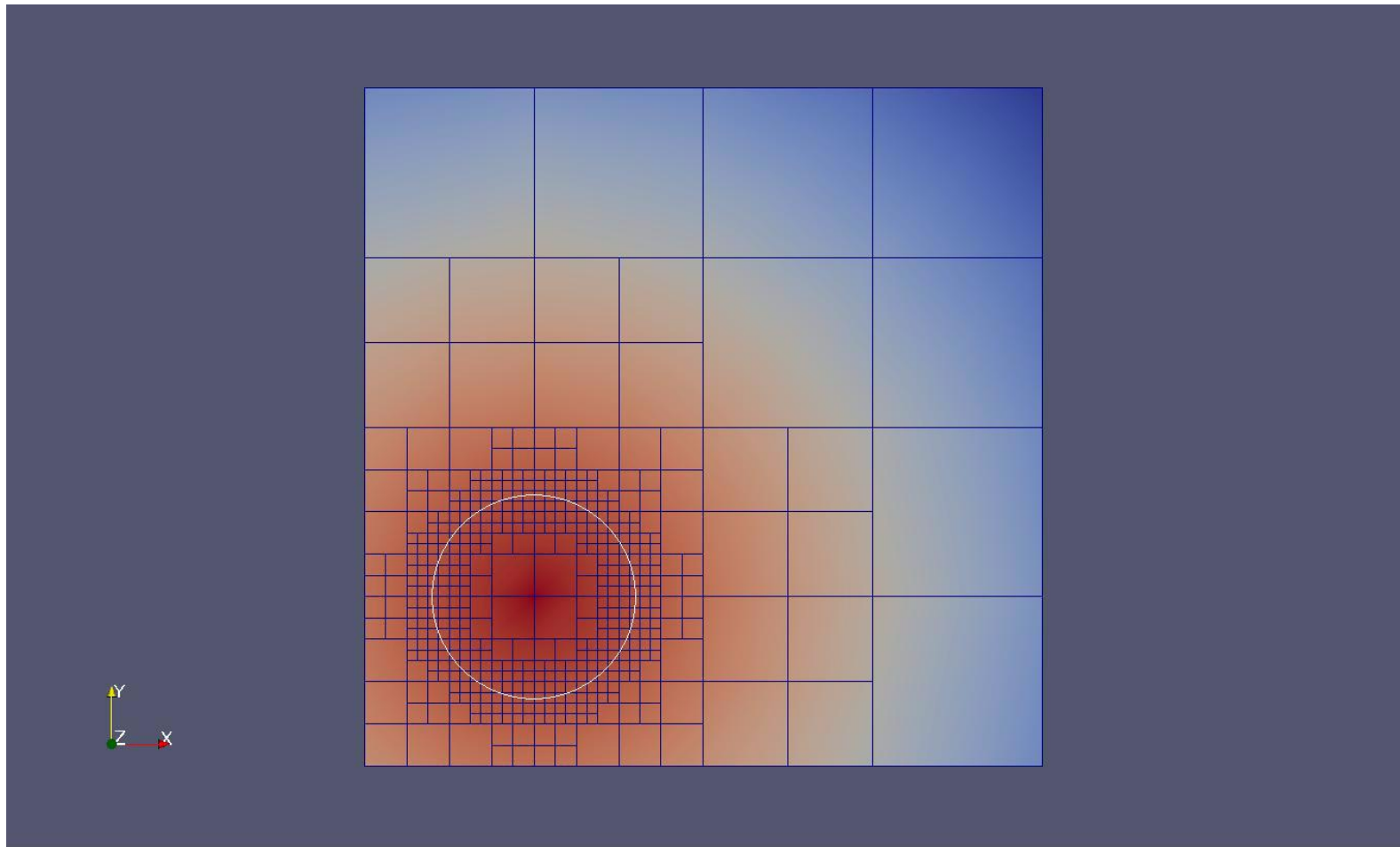
What are Level Sets?



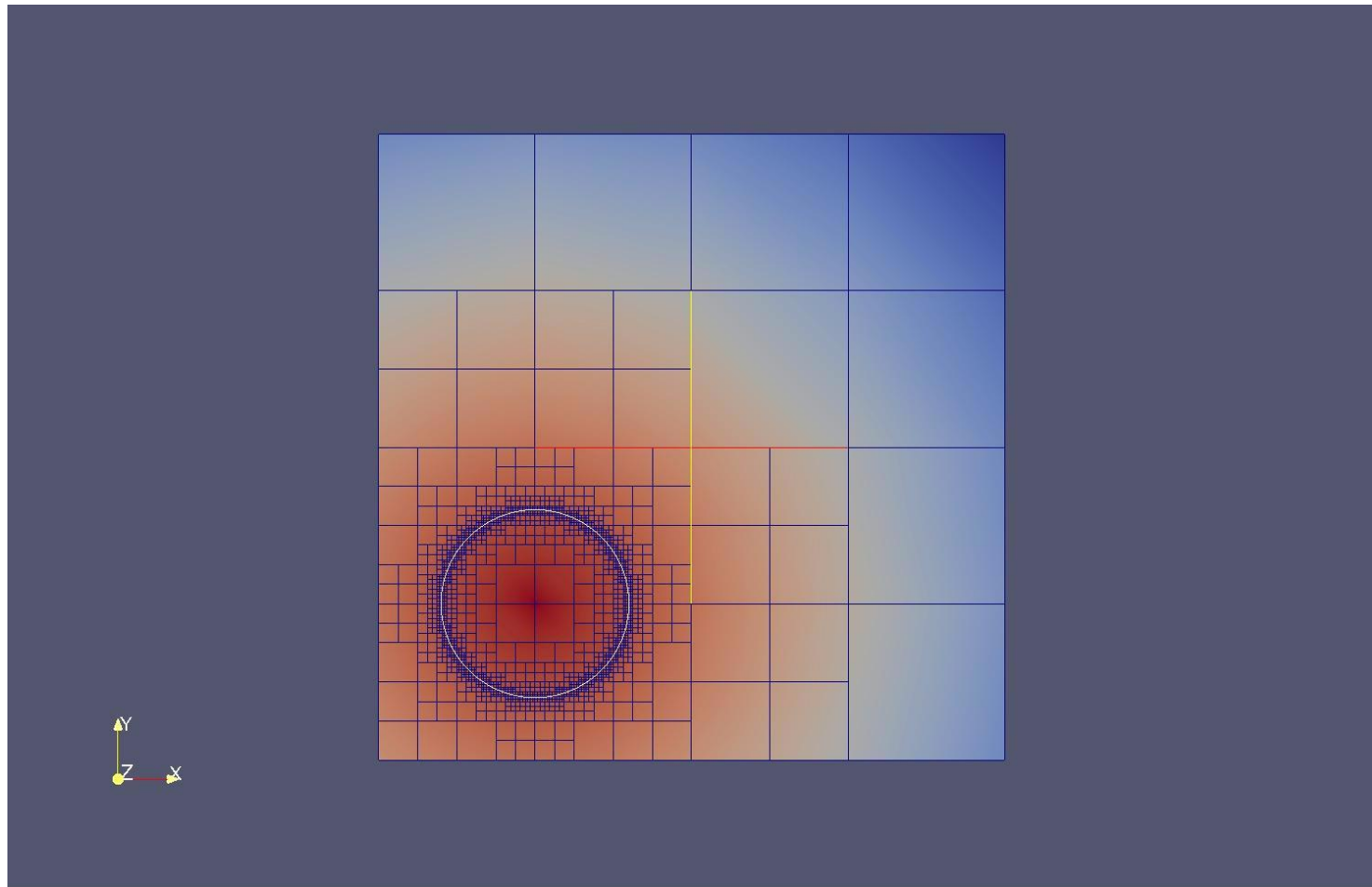
Retrieved from http://en.wikipedia.org/wiki/Level_set_method



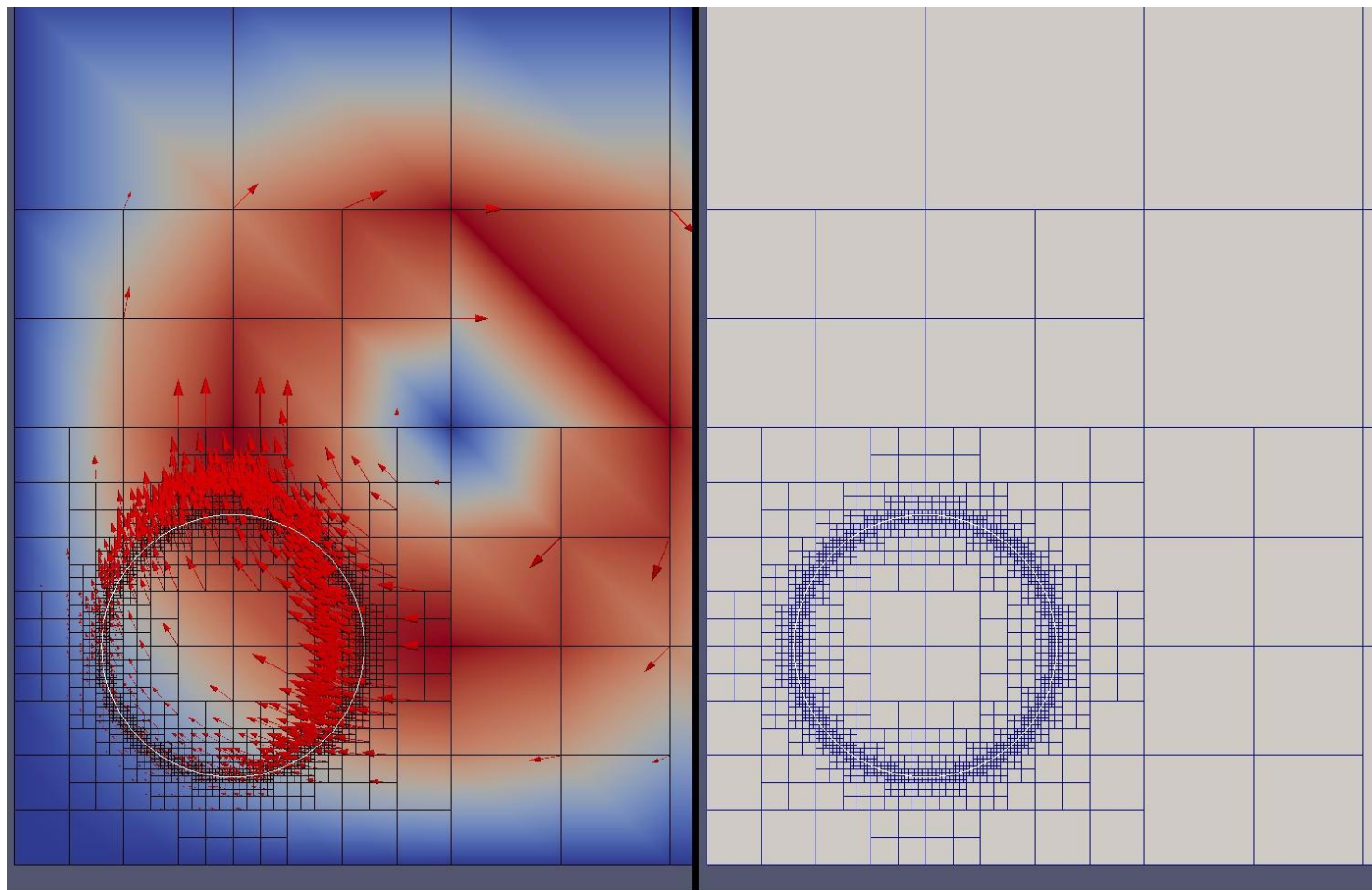
Level Set Simulation



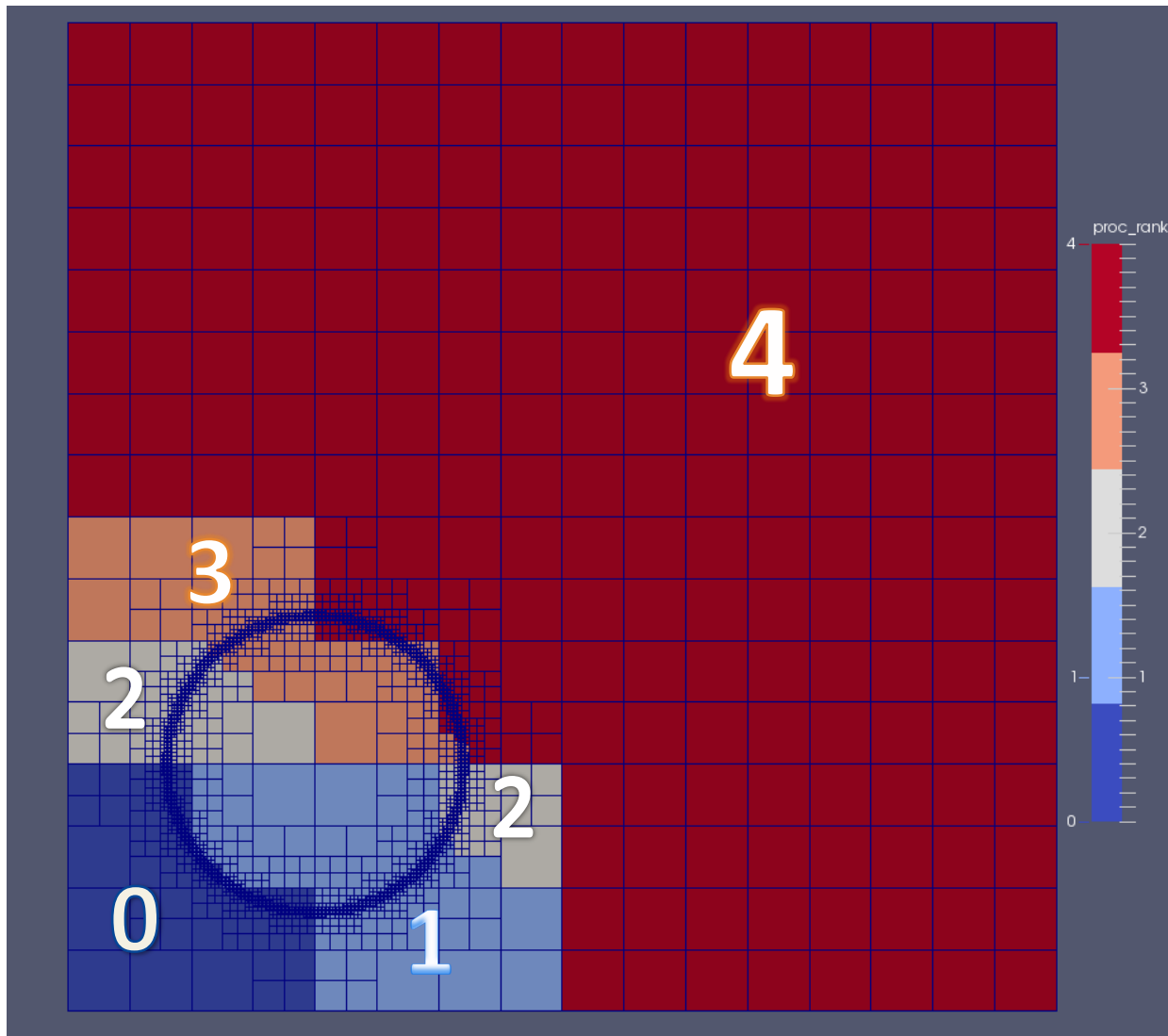
Buggy simulation



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.



Acknowledgments

Mentor: Mohammad Mirzadeh

Faculty Advisor: Frederic Gibou

Institute of Collaborative Biotechnologies

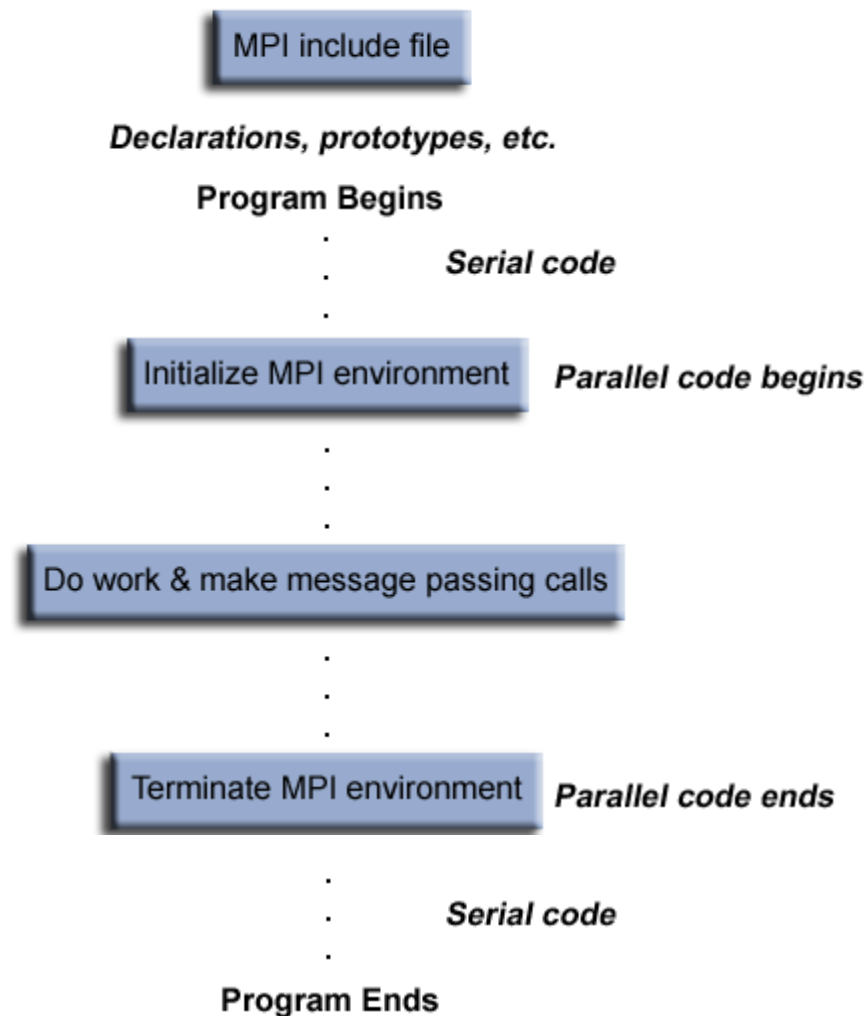
National Science Foundation

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





MPI Code Development



Retrieved from <https://computing.llnl.gov>

Example of MatLab Code

```
1 %Max Rows
2 M = 100;
3
4 %Max Cols
5 N = 100;
6
7 %Start and end points of grid
8 xmin = 0;
9 xmax = 5;
10 ymin = 0;
11 ymax = 5;
12
13 %Init the uniform grid
14 xi = linspace(xmin, xmax, M);
15 yj = linspace(ymin, ymax, N);
16
17 %Init an MxN matrix with all zeros.
18 levelSet = zeros(M, N);
19 phi_dest = zeros(M, N);
20
21 %Radius
22 r = 0.5;
23
24 %Circle levelset at the center of the grid
25 phi = @(x,y) r - sqrt(((x - (xmax/2)).^2) + ((y - (ymax/2)).^2));
26
27 %Side struct for the side lengths of a destination point
28 s_top = 't';
29 s_left = 'l';
30 s_right = 'r';
31 s_bottom = 'b';
32 side = struct(s_top, 0, s_left, 0, s_right, 0, s_bottom, 0);
33
34 %Omega struct for the corner points
35 MM = 'mm';
36 PM = 'pm';
37 MP = 'mp';
38 PP = 'pp';
39 omega = struct(MM, 0, PM, 0, MP, 0, PP, 0);
40
41 %Init various variables for computations
42 vx = 0;
43 vy = 5;
44 dx = (xmax - xmin) / (M - 1);
45 dy = (ymax - ymin) / (N - 1);
46 dt = dx;
47
48 %Calculate the level-set at each point of the grid
49 for row = 1:M;
50     for col = 1:N;
51         levelSet(row, col) = phi(xi(row), yj(col));
52     end
53 end
54
55 %([...%])
56
57 figure;
58 contour(levelSet, [0 0]);
59
60 %Backtracing: calculate a matrix of x_destinations and y_destinations from a given set of points @ pos(x,y)!
61 x_dest = backtrace(xi, vx, dt);
```

```
est_p, dx)
ough 10
3 3 5 6 6 7 9 10
ough 20
12 14 14 16 17 18 19 19
ough 30
23 23 24 26 27 28 28 29
ough 40
33 33 35 36 37 38 38 40
ough 50
43 43 45 46 46 48 48 50
```

workspace

Name	Value
M	100
MM	'mm'
MP	'mp'
N	100
PM	'pm'
PP	'pp'
ans	<1x51 do
col	100
dt	0.0505
dx	0.0505
dy	0.0505
it	15
levelSet	<100x100
omega	<1x1 stru
phi	@(x,y)r-sq
phi_dest	<100x100
r	0.5000
row	100
s_bot	'b'
s_left	'l'
s_right	'r'
s_top	't'
side	<1x1 stru
vx	0
vy	5

Command History

- omega.mm
- proj_backtrace_find
- xi
- x_dest
- x_dest_p
- x_dest
- proj_backtrace_find
- size(x_dest)
- proj_backtrace_find
- x_dest_p
- xi
- side.l
- omega
- omega.mm
- omega.pm
- proj_backtrace_find
- findQuad(1.5, 1)
- proj_backtrace_find
- size M
- M
- size levelSet
- findQuad(x_dest_p, c
- interpolate
- proj_backtrace_find

Example of C++ Code

```
64 int main (int argc, char* argv[]){
65     mpi_context_t mpi_context, *mpi = &mpi_context;
66     mpi->mpicomm = MPI_COMM_WORLD;
67     p4est_t *p4est;
68     p4est_nodes_t *nodes;
69     circle circ(0.5, 0.5, .3);
70     cf_grid_data_t data = {&circ, 7, 0, 1.0};
71     Session::init(argc, argv, mpi->mpicomm);
72     parStopWatch w1, w2;
73     w1.start("total time");
74     MPI_Comm_size (mpi->mpicomm, &mpi->mpisize);
75     MPI_Comm_rank (mpi->mpicomm, &mpi->mpirank);
76     // Create the connectivity object
77     w2.start("connectivity");
78     p4est_connectivity_t *connectivity;
79     my_p4est_brick_t brick;
80     connectivity = my_p4est_brick_new(2, 2, &brick);
81     w2.stop(); w2.read_duration();
82     // Now create the forest
83     w2.start("p4est generation");
84     p4est = p4est_new(mpi->mpicomm, connectivity, 0, NULL, NULL);
85     w2.stop(); w2.read_duration();
86     // Now refine the tree
87     w2.start("refine");
88     p4est->user_pointer = (void*)&data;
89     p4est_refine(p4est, P4EST_TRUE, refine_levelset, NULL);
90     w2.stop(); w2.read_duration();
91     // Finally re-partition
92     w2.start("partition");
93     p4est_partition(p4est, NULL);
94     w2.stop(); w2.read_duration();
95     // generate the node data structure
96     nodes = my_p4est_nodes_new(p4est);
97     // Initialize the level-set function
98     vector<double> phi(nodes->num_owned_indeps);
99     for (p4est_locidx_t i = 0; i<nodes->num_owned_indeps; ++i)
100     {
101         p4est_indep_t *node = (p4est_indep_t*)sc_array_index(&nodes->indep_nodes, i);
102         p4est_topidx_t tree_id = node->p.piggy3.which_tree;
103         p4est_topidx_t v_mm = connectivity->tree_to_vertex[P4EST_CHILDREN*tree_id + 0];
104         double tree_xmin = connectivity->vertices[3*v_mm + 0];
105         double tree_ymin = connectivity->vertices[3*v_mm + 1];
106     }
```

```
1 #include "parallel_semi_lagrangian.h"
2 #include "bilinear_interpolating_function_p.h"
3 #include <src/refine_coarsen.h>
4 namespace parallel{
5     SemiLagrangian::SemiLagrangian(p4est_t **p4est, p4est_nodes_t **nodes)
6     : p_p4est (p4est), p4est (*p4est),
7       p_nodes_(nodes), nodes_(*nodes)
8     {
9         // compute domain sizes
10        double *v2c = p4est->connectivity->vertices;
11        p4est_topidx_t *t2v = p4est->connectivity->tree_to_vertex;
12        p4est_topidx_t first_tree = 0, last_tree = p4est->last_local_tree;
13        xmin = v2c[3*t2v[P4EST_CHILDREN*first_tree + 0] + 0];
14        ymin = v2c[3*t2v[P4EST_CHILDREN*first_tree + 0] + 1];
15        xmax = v2c[3*t2v[P4EST_CHILDREN*last_tree + 3] + 0];
16        ymax = v2c[3*t2v[P4EST_CHILDREN*last_tree + 3] + 1];
17    }
18    double SemiLagrangian::advect(const CF_2 &vx, const CF_2 &vy, std::vector<double>& phi){
19        double dt = compute_dt(vx, vy);
20        p4est_topidx_t *t2v = p4est->connectivity->tree_to_vertex; // tree to vertex list
21        double *t2c = p4est->connectivity->vertices; // coordinates of the vertices of a tree
22        int rank_current = 0, rank_lookup = 0; //P: Used to check for ranks
23        //P: Buffers for local and non-local nodes for MPI_Send/MPI_RECV
24        //std::vector<p4est_indep_t> local_dep_points, non_local_dep_points; May not need this.
25        //P: Hold quadrant information for MPI_SEND/MPI_RECV. Changed to a class of vectors instead of a vector of classes
26        quad_information local_dep_points_info, non_local_dep_points_info;
27        // Hold xy & rank for non-local points
28        non_local_point_buffer nL_buffer;
29        // loop over all nodes
30        std::vector<double> phi_np1(phi.size());
31        //P: Loop through all nodes and separate nodes into local and non-local vectors
32        //P: Local vectors will be used normally (serial).
33        //P: Non-local vectors will need to be handled differently
34        for (p4est_locidx_t ni = 0; ni < nodes->num_owned_indeps; ++ni){ //Loop through all nodes
35            p4est_indep_t *indep_node = (p4est_indep_t*)sc_array_index(&nodes->indep_nodes, ni);
36            p4est_topidx_t tree_idx = indep_node->p.piggy3.which_tree;
37            p4est_quadrant_t *quad;
38            p4est_locidx_t quad_idx;
39            p4est_topidx_t tr_mm = t2v[P4EST_CHILDREN*tree_idx + 0]; //mm vertex of tree
40            double tr_xmin = t2c[3 * tr_mm + 0];
41            double tr_ymin = t2c[3 * tr_mm + 1];
42            //Find initial xy points
43            double xy[] =
44            {
```



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

C / C++

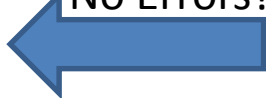


Parallelize code with MPI

Export to visual

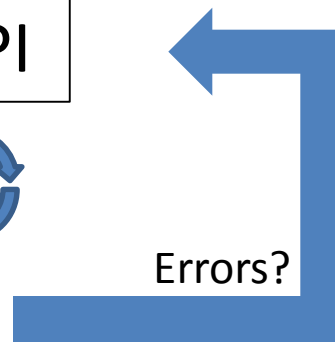


No Errors?



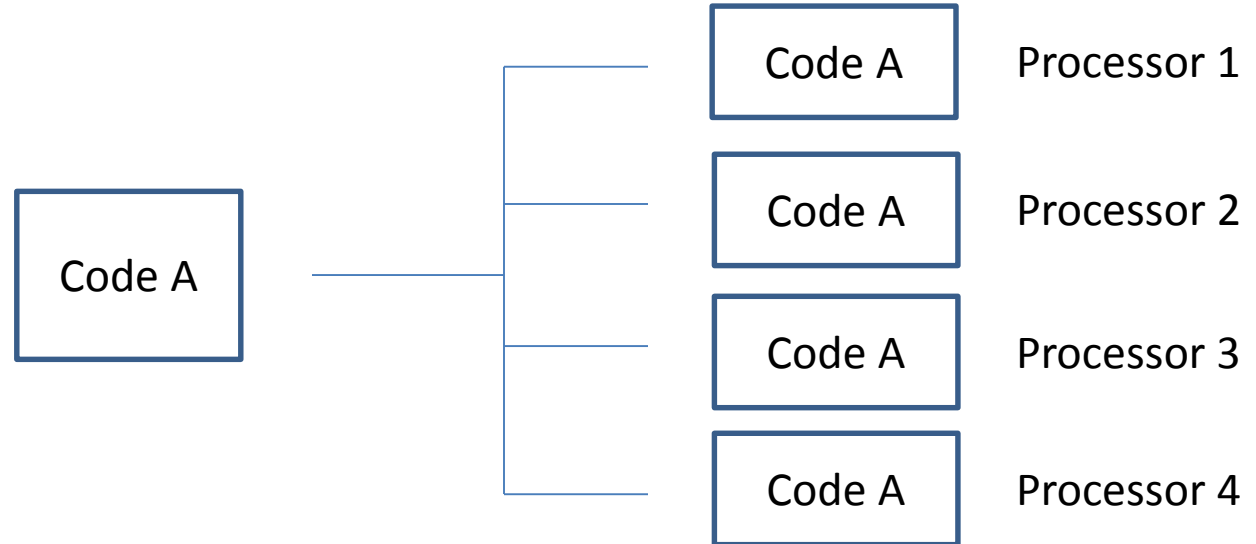
ParaView
Parallel Visualization Application

Errors?

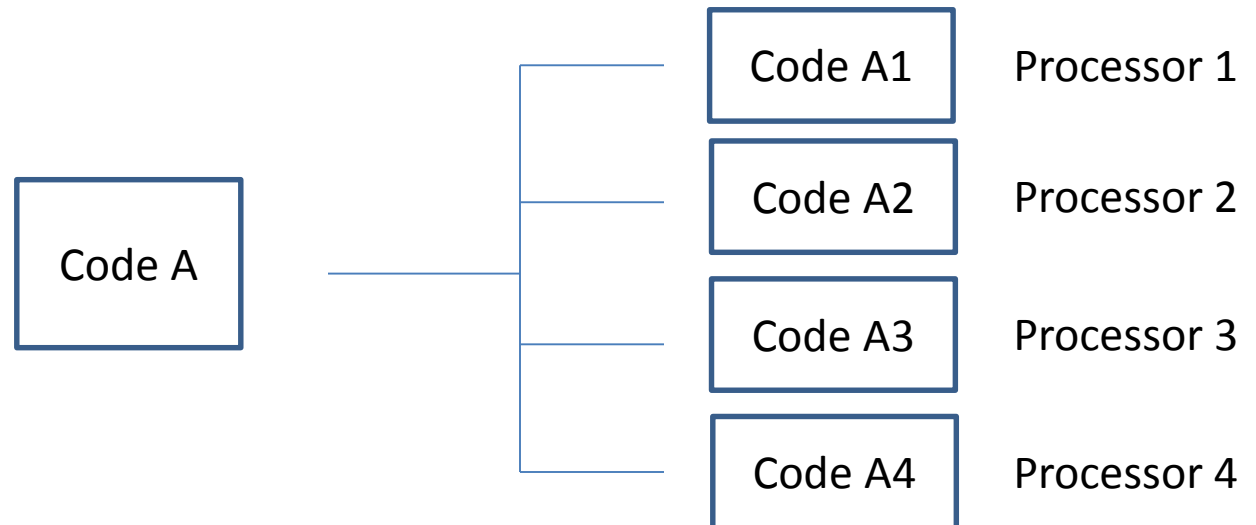


What is the difference between Serial and Parallel programming?

What is Serial Programming?

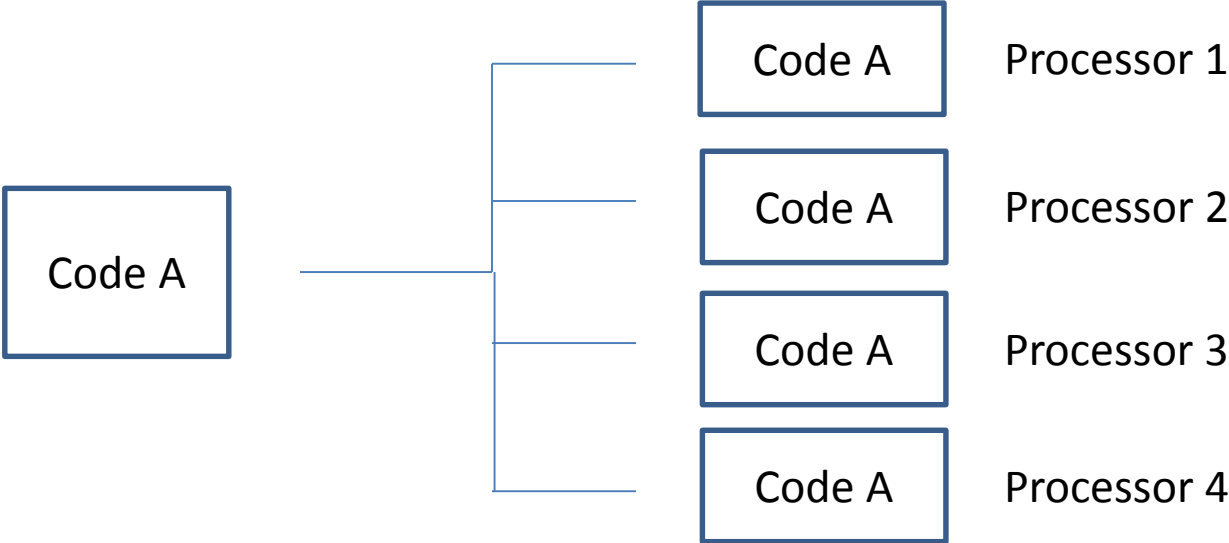


What is Parallel Programming?

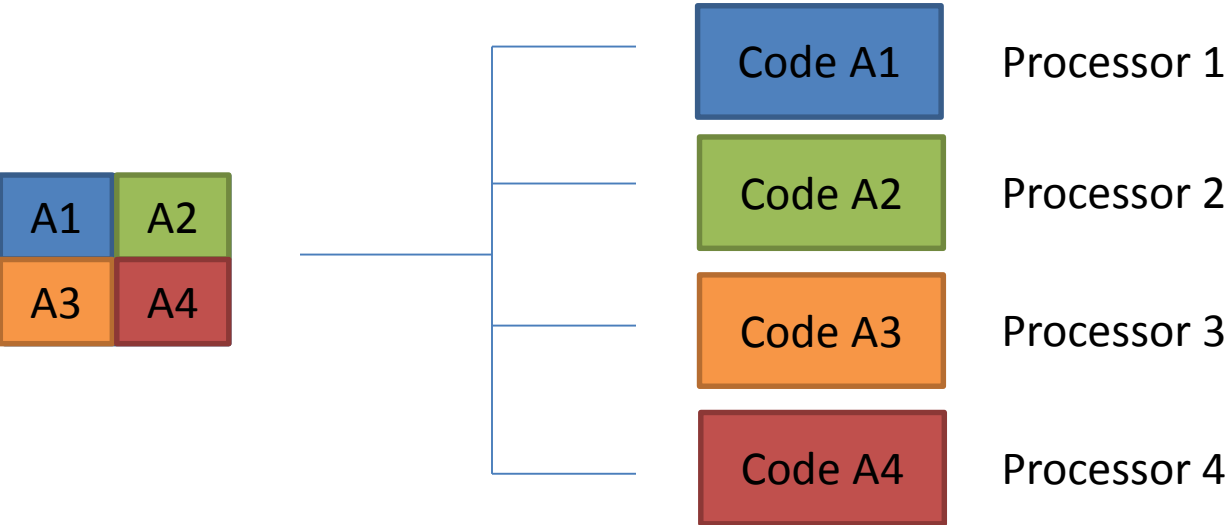


Parallel programming

What is Serial Programming?



What is Parallel Programming?



Filler

- Filler