Applications of Parallel Programming

Chris Kauffman

Last Updated:
Wed Dec 8 09:38:06 AM CST 2021

Logistics

A2 and A3

- Tests updated should resolve Valgrind issues for Problem 1
- Still no Optional Problem 4
- A3 by Wed

Poll on Final Exam

Poll on Canvas concerning Final Exam

- **Option A:** Mini-exam 4 (10%) + Final Exam (10%)
- ~~Option B: Final Exam Last Day of class (20%)~~

Agenda

- Mon: P2 Prob 1 Review of Fluid Dynamics
- Wed: Applications + Mini-Exam 4

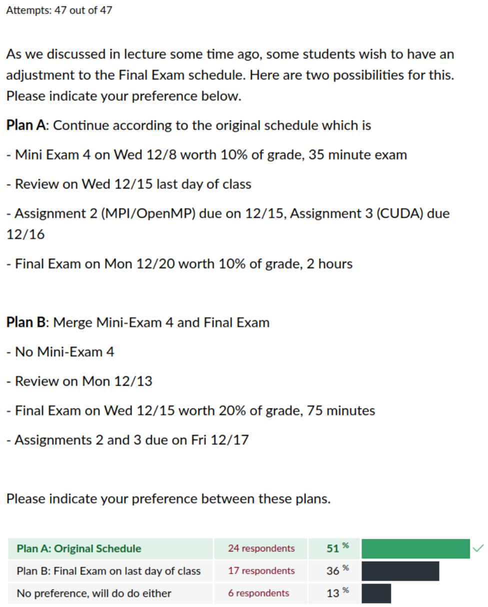
Guest Lecture

Some Applications of Parallel Computing in Fluid Mechanics

Sreevatsa “Sam” Anantharamu, Aerospace Engineering

Final Exam Survey Stats

- Plan on Original Schedule
- Mini-Exam 4 Wed 12/8 on GPUs / CUDA, Fluid Dynamics Application
- Final Exam Mon 12/20 1:30-3:30pm, comprehensive



Attempts: 47 out of 47

As we discussed in lecture some time ago, some students wish to have an adjustment to the Final Exam schedule. Here are two possibilities for this. Please indicate your preference below.

Plan A: Continue according to the original schedule which is

- Mini Exam 4 on Wed 12/8 worth 10% of grade, 35 minute exam

- Review on Wed 12/15 last day of class

- Assignment 2 (MPI/OpenMP) due on 12/15, Assignment 3 (CUDA) due 12/16

- Final Exam on Mon 12/20 worth 10% of grade, 2 hours

Plan B: Merge Mini-Exam 4 and Final Exam

- No Mini-Exam 4

- Review on Mon 12/13

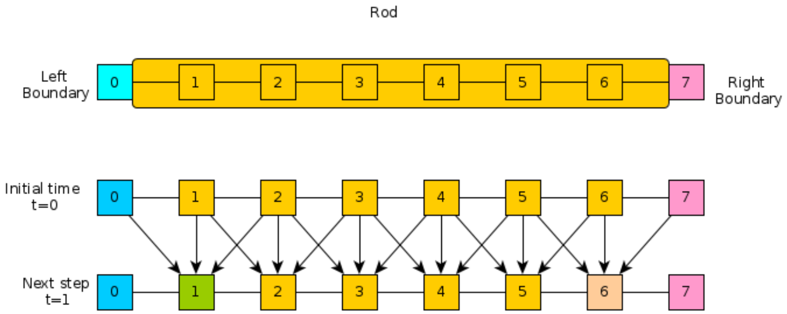
- Final Exam on Wed 12/15 worth 20% of grade, 75 minutes

- Assignments 2 and 3 due on Fri 12/17

Please indicate your preference between these plans.

Plan A: Original Schedule	24 respondents	51 %	✓
Plan B: Final Exam on last day of class	17 respondents	36 %	
No preference, will do do either	6 respondents	13 %	

Note on A2 Problem 1 Heat



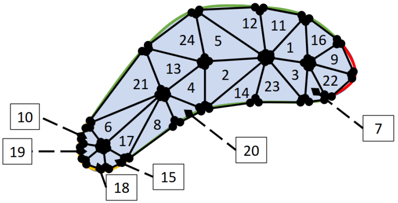
- Assume 4 procs total, Column partition
- Note the need to exchange data between procs
- What MPI calls would be used to exchange
- How does one most efficiently arrange communication?

Avoid "chains" of communication that would block procs for proceeding as quickly as possible.

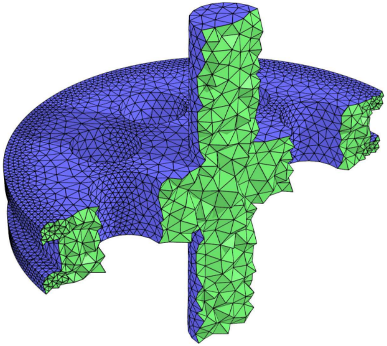
Reviewing Fluid Dynamics: Meshes

Divide the domain into triangles; the set of triangles is called a mesh/grid

Done long in advance of running any simulations



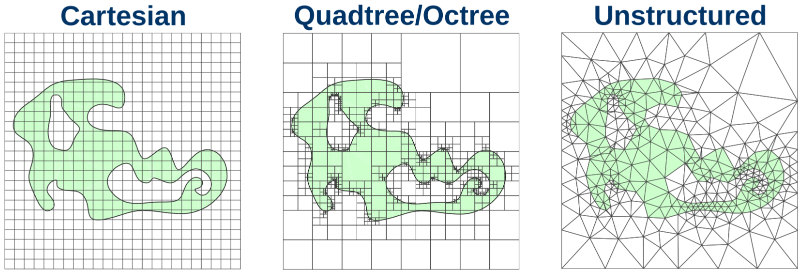
Source: Some applications of parallel computing in fluid mechanics by Sreevatsa Anantharamu



Source: Mesh Generation for Implicit Geometries by Per-Olof Persson (Thesis)

Reviewing Fluid Dynamics: Mesh Generation

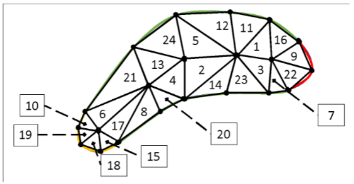
Mesh generation has many styles / techniques, is its own (parallel) problem



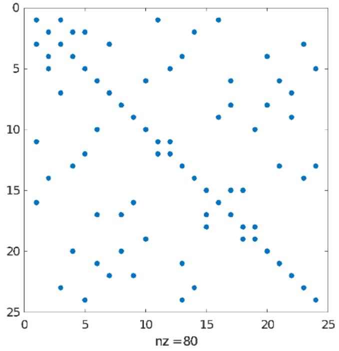
Exercise: Mesh to Matrix

After creating a mesh, Sam alluded to getting a sparse matrix from it.

1. What is this matrix?
2. Why is it sparse?
3. What relation does it have to the Heat problem we have studied several times?
4. What would the matrix for the Heat problem look like?
5. Sam described it as an "unstructured" sparse matrix. What does this mean?



Mesh



Sparsity pattern of A

Answers: Mesh to Matrix

1. What is this matrix?
   Matrix of neighbors: which triangles share a face with others.
2. Why is it sparse?
   Not all triangles share faces. Each triangle only has 3 faces so has at most 3 neighbors.
3. What relation does it have to the Heat problem we have studied several times?
   The heat problem also defined neighbors which dictated heat transfer. Each element in the rod had 2 neighbors (left and right).

Answers: Mesh to Matrix

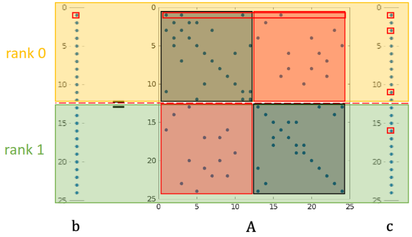
4. What would the matrix for the Heat problem look like?
   Tri-diagonal: element i has neighbors $i-1$ and $i+1$.

$$T = \begin{pmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & c_2 & \ddots & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & c_{n-1} & a_n \end{pmatrix}$$

5. Sam described it as an "unstructured" sparse matrix. What does this mean?
   For an arbitrary mesh, the neighbor matrix does not follow a regular pattern like in the Heat problem. Element 4 has neighbors 2, 13, 20; no pattern for the neighbors.

Exercise: Sparse Matrix Vector Multiply

- Sam indicated a Sparse Matrix/Vector multiply is a central operation for the simulation
- Advocated using a simple Row Partition scheme for the matrix, input vector, output vector



Questions

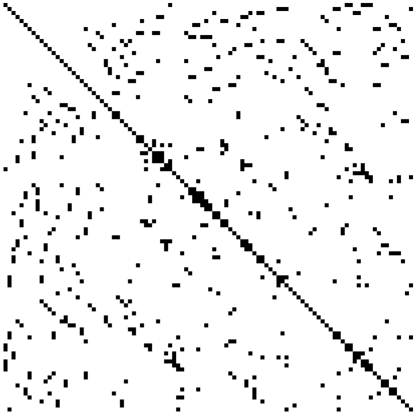
1. What need for communication does this incur?
2. Why is this situation more complex than our work on Page Rank: give 2 reasons?

Answers: Sparse Matrix Vector Multiply

1. What need for communication does this incur?
   Procs do not have the entire RHS vector; must communicate with other procs to multiply it. Other procs must know who will ask for data from them.
2. Why is this situation more complex than our work on Page Rank: give 2 reasons?
   2.1 *Pagerank was dense so the data communicated were just blocks of dense arrays. In sparse problems must communicate in a format like Compressed Sparse Row (CSR).*
   2.2 *We assumed whole RHS vector fits on procs so no need to communicate before multiplying: all procs had their entire own copy.*

Exercise: Distribution of Sparse Matrices

- Given a square sparse matrix with N rows
- Have a parallel computer with P procs
- Will partition RHS vector as well
- Which proc should get which rows (matrix/RHS vector)?
- What criteria should be used to dictate partitioning?



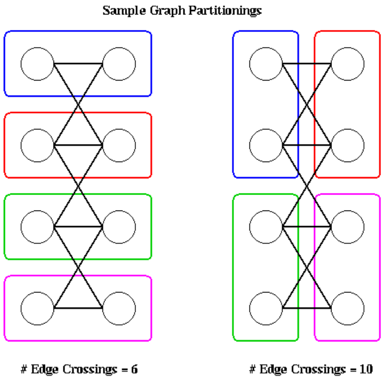
Answers: Distribution of Sparse Matrices

- Want to minimize communication between procs
- If Element 4 has neighbors 2, 13, 20, put rows 2,4,13,20 on same processor
- Leads to a classic NP-Hard problem: **Graph Partitioning**

  In mathematics, a graph partition is the reduction of a graph to a smaller graph by partitioning its set of nodes into mutually exclusive groups. Edges of the original graph that cross between the groups will produce edges in the partitioned graph. If the number of resulting edges is small compared to the original graph, then the partitioned graph may be better suited for analysis and problem-solving than the original.

  – *Wikipedia: Graph Partitioning*

Graph Partitioning



Source: CS267 Lectures 11 and 12 Note by James Demmel, Berkeley

- Graph partitioning seeks a "reduced" graph where multiple vertices are merged into a single vertex group
- Edges within group cost nothing - on same processor
- Seek to minimize edges between groups, the **Cut** of the partition
- Widely used approach is METIS (serial) and ParMETIS (parallel) by our own George Karypis and Vipin Kumar

Next Time: Brief other Applications

- Molecular Dynamics
- Neural Networks
- Cryptocurrency Mining / Password Cracking

N-Body Simulations

Several types of Physics and Chemistry systems are simulated in the following way

1. N particles exist in some space
2. A **Force Field** is present which describes attraction/repulsion between particles (and possibly external forces)
3. Particles have their positions initialized in the space
4. Forces acting on each particle are calculated using the Force Field and the current state (position/velocity) of the particles
5. The positions of each particle are adjusted by a small amount based on forces acting on them
6. Repeat 4/5 until bored

In Astrophysics, particles are usually celestial bodies (planets, stars, galaxies) and Force Field is Gravity.

Exercise: Sample N-Body

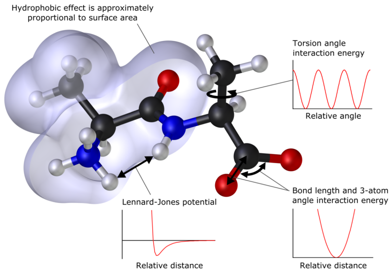
```
# calculate forces for new accelaration
for i=0 to N-1:
  acc[i].x = 0
  acc[i].y = 0
  for j=0 to N-1:
    if i==j: continue
    dx = pos[j].x - pos[i].x
    dy = pos[j].y - pos[i].y
    inv_r3 = (dx*dx + dy*dy)**(-1.5);
    acc[i].x +=  G * (dx * inv_r3) * mass[j];
    acc[i].y +=  G * (dy * inv_r3) * mass[j];

# calculate new position
for i=0 to N-1:
  vel[i].x += acc[i].x
  vel[i].y += acc[i].y
  pos[i].x += vel[i].x * dt
  pos[i].y += vel[i].y * dt
```

- Code at the right represents (very rough) N-body simulation in 2D
- What opportunities do you see for parallelism?

Molecular Dynamics

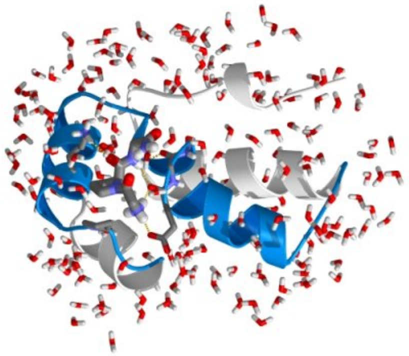
- Chemistry simulations can use a similar approximation to N-body but are focused on microscopic systems: particles are Atoms and Force Field is electrical interactions
- **Bonds** between atoms which act like springs which stretch/compress and allow twisting
- **Van der Waals** forces between non-bonded atoms which exerts attractive/repulsive forces


Hydrophobic effect is approximately proportional to surface area
Torsion angle interaction energy
Relative angle
Bond length and 3-atom angle interaction energy
Lennard-Jones potential
Relative distance
Relative distance

Source: Wikip Molecular Mechanics

Protein Folding

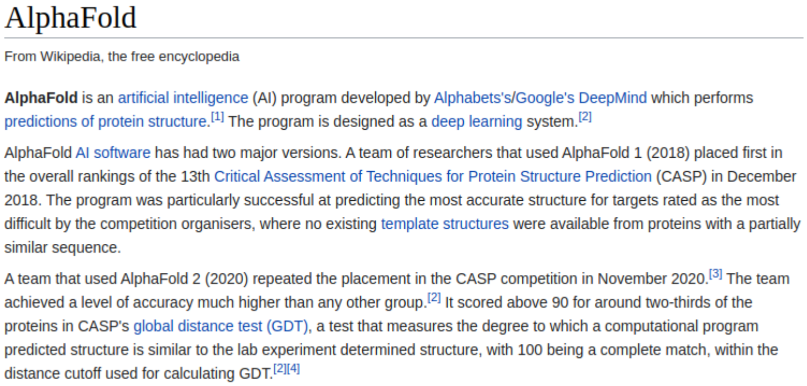
- Protein Folding is one heavily studies area of Molecular Dynamics
- Through experimental techniques / genetics can determine the bonded atoms a protein: 1,000-25,000 atoms with water around it
- Proteins are a large linear molecule that "folds" due to interactions with surrounding water + self attraction



Source: Intro to protein simulations

Segue to Machine Learning

- Protein Folding is a tough physic simulation
- Has led to various approximations, heuristic, computational approaches. Notably, recent late Deep Learning approaches...



AlphaFold

From Wikipedia, the free encyclopedia

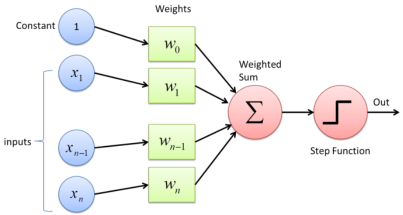
AlphaFold is an artificial intelligence (AI) program developed by Alphabets's/Google's DeepMind which performs predictions of protein structure.[1] The program is designed as a deep learning system.[2]

AlphaFold AI software has had two major versions. A team of researchers that used AlphaFold 1 (2018) placed first in the overall rankings of the 13th Critical Assessment of Techniques for Protein Structure Prediction (CASP) in December 2018. The program was particularly successful at predicting the most accurate structure for targets rated as the most difficult by the competition organisers, where no existing template structures were available from proteins with a partially similar sequence.

A team that used AlphaFold 2 (2020) repeated the placement in the CASP competition in November 2020.[3] The team achieved a level of accuracy much higher than any other group.[2] It scored above 90 for around two-thirds of the proteins in CASP's global distance test (GDT), a test that measures the degree to which a computational program predicted structure is similar to the lab experiment determined structure, with 100 being a complete match, within the distance cutoff used for calculating GDT.[2][4]

Perceptrons

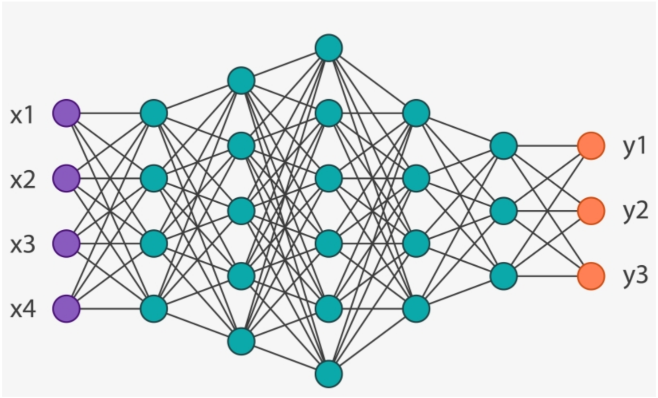
- Neural networks are comprised of a series of **Perceptrons**
- Input data vector is multiplied by a Weight vector and summed
- Basically a dot product followed by a (somewhat) arbitrary Activation function
- In a Single Perceptron, there is a vector of weights that are learned through gradient descent to predict an output value



Source: "What the Hell is a Perceptron" by Towards Data Science

- Will focus on Parallelism in computing outputs for neural networks - also parallelism in **backpropagation** to learn weights

Exercise: Neural Networks and Parallelism



- Neural nets are layers of Perceptrons connected in some pattern: outputs in one layer become inputs in another
- Where can one find parallelism when calculating inputs to outputs in a neural net?

GPU Acceleration of Neural Nets

- Evaluating a layer boils down to a Matrix-Vector multiply then applying a function to each output vector element
- Matrix-Vector multiply is a well-studied parallel problem...
- Neural nets can be done on distributed machines but it is more common to use single machines with multithreading
- Since GPUs fit the problem extremely nicely and are often used to accelerate Neural Network training and predictions
- Popular packages for neural networks like
  - TensorFlow (by Google)
  - Pytorch (based on Facebook's Torch library)

  all include GPU support for accelerating neural network operations