Peachy Parallel assignments
https://tcpp.cs.gsu.edu/curriculum/?q=peachy

• Tested

• Adoptable

• Cool and inspirational
Using MPI For Distributed Hyper-Parameter Optimization and Uncertainty Evaluation

John Li, Erik Pautsch, Silvio Rizzi, Maria Pantoja, and George K. Thiruvathukal,
Goal Accelerate Uncertainty Evaluation in AI
How to Accelerate Uncertainty Evaluation

```python
# n is # ensembles, size is the # of MPI nodes
count = n // size
# extra catchments if n is not a multiple of size
remainder = n % size
# processes with rank < remainder analyze one extra catchment
if rank < remainder:
    start = rank * (count + 1) # index of first catchment to analyze
    stop = start + count + 1 # index of last catchment to analyze
else:
    start = rank * count + remainder
    stop = start + count
if rank > 0:
    comm.Send(...) # send to leader node
else:
    final_results = .. # final results printed by the leader node
```
Code And Slides

The link for the above assignment can be found https://drive.google.com/drive/folders/1KrxWIMZpoJzph0Y7VbZj_yYyACK-Jusl?usp=sharing

Thanks to:

**Sustainable Horizons Institute** which is part of the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration and by Argonne National Laboratory.

This research used resources of the **Argonne Leadership Computing Facility**, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.
Solving the 1D Heat Equation in Chapel

Jeremiah Corrado
Assignment Summary
Background and Algorithm

1D Heat Equation: \[ \frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \]

Finite-Difference Heat Equation: \[ u^{n+1}_i = u^n_i + \alpha (u^n_{i-1} - 2u^n_i + u^n_{i+1}) \]

Finite Difference Algorithm:
- define \( \Omega \) to be a set of discrete points along the x-axis
- define \( \hat{\Omega} \) over the same points, excluding the boundaries
- define an array \( u \) over \( \Omega \)
- set some initial conditions
- create a temporary copy of \( u \), named \( un \)
- for \( N \) timesteps:
  1. swap \( u \) and \( un \)
  2. compute \( u \) in terms of \( un \) over \( \hat{\Omega} \)

```plaintext
1 const omega = {0..<nx},
2 omegaHat = omega.expand(-1);
3 var u: [omega] real = 1.0;
4 u[nx/4..3*nx/4] = 2.0;
5 var un = u;
6 for 1..N {
7    un <=> u;
8    forall i in omegaHat do
9       u[i] = un[i] + alpha * 
10          (un[i-1] - 2*un[i] + un[i+1]);
11 }
Assignment Summary
Distributing a parallel program

1. Start with a simpler data-parallel program
   - Provide students with examples of using distributed arrays in Chapel
   - Ask students to modify the data-parallel program to use distributed arrays

2. Start with a lower-level task-parallel program
   - Provide students with examples of controlling the locality of task execution in Chapel
   - Ask students to modify the task-parallel program to execute tasks across multiple compute nodes
Key HPC Concepts Covered

Parallelizing order independent loops

Locality of data and computation

Barriers and synchronization

Inter-node communication
Key HPC Concepts Covered

Parallelizing order independent loops

forall i in omegaHat do
  u[i] = un[i] + alpha *
  (un[i-1] - 2*un[i] + un[i+1]);

Locality of data and computation

const omega = Block.createDomain({0..<nx});
var u : [omega] real;

coforall tid in haloDist do
  on tid.locale do
    taskSimulate(tid);

Barriers and synchronization

var b = new barrier(nTasks);
...
for l..nt {
  ...
  b.barrier();
  uLocal1 <=> uLocal2;
  ...
}

Inter-node communication

if tid != 0 then
  halos[tid-1][RIGHT] = uLocal2[omegaLocal.low];
if tid != nTasks-1 then
  halos[tid+1][LEFT] = uLocal2[omegaLocal.high];
Summary

• An introductory HPC assignment that uses a practical problem to teach several concepts:
  • parallelism, synchronization, locality, communication

• Leverages Chapel’s first-class notions of parallelism, locality and distributed arrays
  • less focus on the software engineering
  • more focus on the HPC concepts themselves

• Students are asked to do the same thing in two different ways (data parallel & task parallel)
  • repetition helps cement fundamental concepts
  • exposes students to multiple perspectives on the same problem
Q &A

Resources:
Github Repo for assignment: https://github.com/jeremiah-corrado/Chapel-Heat1D-PPA
Chapel Homepage: https://chapel-lang.org/
Chapel Blog: https://chapel-lang.org/blog/

Contact:
email: jeremiah.corrado@hpe.com
chapel discourse: https://chapel.discourse.group/
**MapReduce MPI**

- Developed by K. Devine and S. Plimpton at Sandia
- Essentially a distributed hash table processing engine
- Sit atop MPI
- Used for data processing in MPI codes
- Will do out of core if necessary
- If you teach MPI, it’s easy to teach MapReduce

---

**k-NN**

- $N$ categorized points in $d$ dimensions
- $q$ query points
- For each query points
  - Find the $k$ closest points
  - Vote to guess the category
**$k$ Nearest Neighbor in MapReduce MPI**

Rough solution

- All processes read queries
- Map the datapoints files in parallel to generate \((\text{query}, (\text{dist}, \text{class}))\) pairs
- Reduce per query to get \((\text{query}, (\text{dist}_1, \text{class}_1, \ldots, \text{dist}_k, \text{class}_k))\)
- Map to get \((\text{query}, (\text{pred}_1, \text{count}_1, \text{pred}_2, \text{count}_2, \ldots))\)
- Dump to output

Some optimization:

- $O(nq)$ computation
- Reduce causes $O(nq)$ comm
- Local reduce gives comm in $O(qkP)$

**Thoughts**

- Non trivial application of Map Reduce
- Reinforces locality
- Tons of data available
- Possible optimization to prevent $O(nq)$ calculations
- Can be adapted to MPI for python
- Can be adapted for hybrid MPI-OpenMP
- Can be adapted in Data Structures
Parallelizing a 1-Dim Nagel-Schreckenberg Traffic Model

Ramses van Zon (SciNet HPC, UofT)  Marcelo Ponce (Comp. & Math. Sciences, UTSC)

EduHPC-23, SC23, Denver

November 13, 2023
Peachy Assignment

- The Nagel-Schreckenberg traffic model is a simulation using pseudo-random numbers.
- A serial starter code in C++ is provided.
- Task:
  - Parallelize with OpenMP.
  - Do so in a reproducible way: output has to be independent of number of threads.
  - Aim for good strong and weak scaling.

Model

- Cars have discrete positions and velocities on a circular road.
- At discrete time steps, for each car:
  - Speed-up: If velocity \( v < v_{\text{max}} \), increase \( v \) by one.
  - Avoid collision: If \( v \) would lead to a collision with car in front, reduce \( v \).
  - Randomly break: With given probability \( p \), reduce \( v \) by one.
  - Drive: Move car forward by \( v \) steps.
Nagel-Schreckenberg traffic model results

$L=1000$ $T=1000$ $N=200$ $p=0.13$ $v_{max}=5$ $seed=13$ $per=1$ $outputprefix=test$
Crux of the solution

PRNG are generated serially but some PRNG allow $\log(n)$ skip-ahead.

- **Starter code and assignment description**: https://github.com/Practical-Scientific-and-HPC-Computing/Traffic_EduHPC-23
Program Your Favorite Data Science Pipeline

- ~ 3 data analyses on ~ 2 datasets
- in teams (~ 3 students)
- 3 weeks
- presentation of results
- submission of report and executable code
Data Science Pipeline – NYC Crime

Data Aggregation

<table>
<thead>
<tr>
<th>Polygon</th>
<th>Area</th>
<th>NTA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Population</th>
<th>Age</th>
<th>NTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>Felony</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td>Violation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2020</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cleaning, Filtering & Analysis

Visualization & Presentation

https://git.uni-jena.de/big_data_assignments/projects
Assignment Evaluation

- high degree of freedom in task realization
- ratio supervisors and students
- fair grading
Assignment Evaluation

- high degree of freedom in task realization
- high motivation
- ratio supervisors and students
- independent and creative thinking
- fair grading
- team building
Assignment Evaluation

- High degree of freedom in task realization
- Ratio supervisors and students
- Fair grading
- High motivation
- Independent and creative thinking
- Team building

Practical Experience with Big Data
Find our NYC Crime example at ...

https://git.uni-jena.de/big_data_assignments/projects

Thank you for your attention!

Marieke Plesske
K-Means: An assignment for OpenMP, MPI and CUDA/OpenCL

Diego García-Álvarez, Arturo Gonzalez-Escribano

Trasgo Group, University of Valladolid, Spain

EduHPC’2023
Nov 13th, 2023
Context

- Different parallel programming models
  - Different approaches for parallelizing the same problem
  - Understand the differences is key
  - Needed in modern heterogeneous systems

- Target: Parallel Computing course
  - Computer Engineering degree, 3rd year, Major elective
  - Three practical programming blocks: OpenMP, MPI, CUDA

- Teaching methodology:
  - Based on projects
  - Competitive + Collaborative gamification

- Series of peachy assignments used for the contest activity:
  EduHPC'18, '19, '20, '21, '22
Assignment objectives

- Use the same example program in the three blocks
- Show portability of different key parallelization approaches and techniques
- Observation: Large gap between examples of programming primitives/structures and complex contest codes
- This year: A simpler assignment, focus on basic concepts and their portability
- Students start with:
  - Handout
  - Sequential code with the part to parallelize clearly marked
  - Some examples of input arguments (more can be easily generated)
K-means clustering

- Powerful and popular data mining algorithm:
  Segmentation, pattern analysis, image compression, etc.
- Split a cloud of n-Dimensional points in clusters with minimum distance to a centroid
  - Init: Read points, randomly fix centroid positions
  - Main clustering loop
    - Re-assign points to the nearest centroid
    - Compute new centroid locations:
      Arithmetic mean of assigned points
      (until few re-assignments or max. iterations)

\[ d = \sqrt{(x_{12} - ce_{12})^2 + (x_{12} - ce_{12})^2} \]
Approach and concepts covered

- Previous educational approaches for OpenMP, MPI, and/or CUDA:
  - Skip to parallelize the computing of new centroid locations (load-balance problems)
  - Use dynamic buffers for cluster points

- Our approach:
  - Parallelize all stages; static data structures (simple to manage, easier to debug)
  - Parallelization strategy provided: Help students to apply theory systematically
    - Loop parallelization
    - Solve write and update race conditions: Critical regions, atomics, reductions
    - Basic collective operations and communications, distributed reduction
    - Thread-blocks, coalesced memory access
    - Reduction porting and evaluation
    - Advanced students: Locality optimizations, load balancing problems, ...
Using the assignment

Course and students:
- Students background: O.S. and concurrency, C programming
- 48 students enrolled, working in small teams (2 people)
- One week time for the solution on each model

Tools:
- Modern C compiler with OpenMP, any MPI library, CUDA or OpenCL toolkit
- Code output can be automatically checked for correctness: Tablon
- Better a shared platform for students to compare and discuss results
- In our case: AMD server 64 cores + Intel servers 12 cores, 32 cores + 4 NVIDIA CUDA 3.5 GPUs
Results

- Lower complexity than previous peachy assignments: Lower number of test submissions to the cluster
- Personal interview for each block + survey at the end of the course
- All students agreed that the project improves the concepts understanding
- For the first time: 60% students prefer MPI over OpenMP !
- Solving race conditions is always nasty, Collective communications + static data structures are easy