Parallelizing a Particle Filter Implementation for a Two Dimensional Point Vortex Fluid Model

Adam Mallen and Erin Soderberg
Parallel and Distributed Systems
Marquette University
May 2, 2011
Background

- Data provided by ocean drifters gives quantitative information about the dynamics of the underlying flow.
- Assimilating this data into flow models is useful, but computationally expensive.
- We rework a computationally expensive particle filter simulation to run in parallel on the Pere cluster.
Point-Vortex Model
Data Assimilation

- Data assimilation is the process of taking noisy partial observations of a system and using them to make inferences about the underlying dynamics.
- In our project, we measure (with noise) the position of a tracer and use data assimilation to estimate the position and strength of the two vortices.
Particle Filtering

- Particle filtering is a specific data assimilation technique, which makes no assumptions about the linearity of the model.
- PF uses a large number of random samples as a discrete approximation of the densities of the state variables.
- PF can be computationally unwieldy because they require a large number of samples to approximate high dimensional densities.
Particle Filtering, continued
Concept Design

- **Program run**: M experiment runs with different parameters
- **Experiment**: loads in parameters and runs N trials with given parameters
- **Trial**: generates one “truth” realization of the system and runs an implementation of the particle filter, with P particles in the cloud and T time steps
- **Prediction**: evolves a single particle forward one time step
- **Update**: modifies a particle's weight according to the likelihood of yielding current observation
- **Resampling**: keeps only top 10% of particles with highest weights
Program Run

Directory and input file setup

Experiment 1
Experiment 2

... Experiment M

Directory and output file clean up
Load system parameters and initial conditions

Trial 1

Trial 2

. . .

Trial N

Combine resulting statistics

Save statistics to output files
Trial

Generate 'truth' run

Initialize particle filter variables

Time Step 1

Prediction

Particle 1
Particle 2
... Particle P

Update

Particle 1
Particle 2
... Particle P

Resampling

... Resampling

Time Step T

Prediction

Particle 1
Particle 2
... Particle P

Update

Particle 1
Particle 2
... Particle P

Return error statistics
Implementation

- Each trial run is contained in a Matlab script.
- Each trial script is called by a wrapper bash script.
- Then, this script is used as the executable for the Condor scheduler.
- Finally, we implement a post-processing bash script to combine the results from each trial into statistics for the entire experiment.
Results

- For 50 trials, $t = 3$ minutes in parallel, $t = 113$ minutes in serial
  - Speedup = 37.7
- For 500 trials, $t = 40$ minutes in parallel, $t = 26$ hours in serial
  - Speedup = 39
Conclusion

• Running the simulations in parallel GREATLY reduces the computation time of each experiment.

• The significantly reduced computation time allows researchers to test new filtering techniques and debug code changes quickly.
Questions?

Thanks!