High Performance Computing Systems (CMSC714)

# Lecture 23: Parallel Discrete-event Simulation

Abhinav Bhatele, Department of Computer Science



UNIVERSITY OF MARYLAND

# Announcements

- Project demos: May 6 and 11
- Final project due on: May 13 11:59 pm AoE

# Summary of last lecture

- *n*-body problem: gravitational forces on celestial bodies
- Several parallel algorithms:
  - Barnes-Hut
  - Fast Multipole Method
  - Particle Mesh
  - P3M
- Simulation codes: FLASH, Cello, ChaNGa, PKDGRAV

# Discrete-event simulation

- Modeling a system in terms of events that happen at discrete points in time
- Either model discrete sequence of events
- Or model time-stepped sequences
- Simulation typically involves system state, event list and a global time variable

# Parallel discrete-event simulation

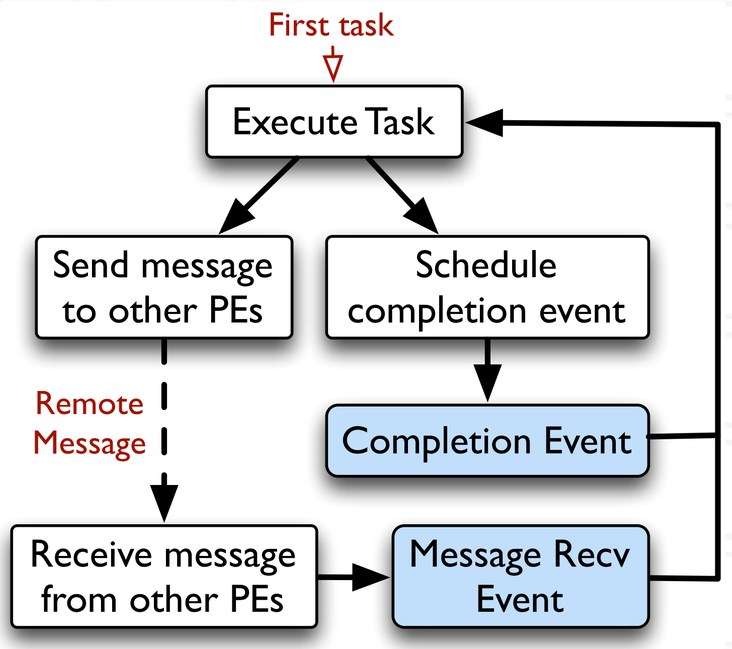
- Divide the events to be simulated among processes
- Send messages wherever there are causality relationships between events
- Synchronize global clock periodically

# Conservative vs. optimistic simulation

- Conservaties DES
  - Do not allow any causality errors
- Optimistic DES
  - Allow causality errors and rollback if needed

# Trace-driven network simulation

- Task is started at time $t_s$
- Possible remote messages to other PEs
  - Kick off other tasks that depend on a message
- Completion event scheduled for time $t_s + t_e$



# Running TraceR in optimistic mode

- Record extra information during forward execution to enable rollback later
  - List of tasks triggered by a message recv or completion event
- Implement reverse handlers for each event

# Epidemiology simulations

- Agent-based modeling to simulate epidemic diffusion
- Models agents (people) and interactions between them
- People interact when they visit the same location at the same time
- These "interactions" between pairs of people are represented as "visits" to locations
- A bi-partite graph of people and locations is used

# EpiSimdemics: Parallel implementation

- All the people and locations are distributed among all processes
- Computation can be done locally in parallel
- Communication when sending visit and infection messages
- Uses Charm++, a message-driven model

```
1  while d ≤ d_max do
2      for p ∈ P do
3          Evaluate scenario trigger conditions;
4          Update health state h_p, if necessary, and reevaluate triggers;
5          foreach v ∈ V_p ( visit schedule of p) do
6              Send visit message m to location l;
7          end
8      end
9      for l ∈ L do
10         foreach m destined for l do
11             Determine the sublocation l_s to visit;
12             Create an arrival and departure event for each visit;
13             Put the events into the event queue q_e of l;
14         end
15         Reorder q_e by the time of event in ascending order;
16         foreach e ∈ q_e do
17             if e is arrival then
18                 Put p into sublocation l_s;
19             else
20                 Remove p from sublocation l_s;
21                 foreach p' currently in l_s do
22                     Compute disease transmission probability q
                         between p' and p;
23                     if q > threshold then
24                         Send infection message to the infected
                             person (p or p');
25                     end
26                 end
27             end
28         end
29     end
30     d++;
31 end
```

# Questions?

UNIVERSITY OF
MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu