TRAFFIC SIMULATION USING MULTI-CORE COMPUTERS

CMPE-655
Adelia Wong, Computer Engineering Dept
Balaji Salunkhe, Electrical Engineering Dept
TABLE OF CONTENTS

- Introduction
- Distributed Urban Traffic Simulator (DUTS)
  - Time-flow Mechanism
  - Levels of detail
- Traffic Models
- Traffic Network
- Decomposition of Distributed System
- Distributed Execution
  - Communication protocols
  - Semi-Centralized Vehicles
- Decomposition of Parallel/Distributed System
- Parallel/Distributed Execution
- Testing Parameters
- Results
  - Quality of Parallelization
  - Comparison of Distributed vs Parallel/Distributed
- Conclusion
- References
INTRODUCTION

- Distributed Urban Traffic Simulator performed on cluster of computers by using two versions, under which various tests have been performed
  - Distributed
    - Utilizes number of single-threaded simulation processes on an equal number of sub-networks
    - Number of processes correspond to number of processor cores of the cluster
    - Inter-process communication performed by using message passing environment
  - Parallel/Distributed
    - Utilizes a number of multi-threaded processes on a lower number of sub-networks
    - Number of processes corresponds to number of computers of the cluster and number of threads per process corresponds to the number of processor cores of each computer
    - Inter-process communication performed by using message passing environment
    - Processes on the same computer communicate using shared memory address

- Road traffic simulation performed on cluster of computers with multi-core processors.
- To get output like real traffic simulation should be very detailed and computation consuming
DISTRIBUTED URBAN TRAFFIC SIMULATOR (DUTS)

- Originally developed at Department of Computer Science and Engineering at University of West Bohemia (DSCE UWB)
- Parallel/distributed simulator was created by an adaptation of DUTS
- Basic features
  - Time-flow mechanism
  - Variable level of detail of road traffic simulation
  - Variable traffic models
TIME-FLOW MECHANISM

- Event step
  - Simulation divided into sequence of incremental actions with associated time-stamp
  - Simulation complete when there are no more events to process

- Time step
  - Simulation is divided into equally sized time steps
  - Simulation is complete when preset number of time-steps is reached
LEVELS OF DETAIL

- **Macroscopic**
  - Low level of detail
  - Traffic flows only
    - mean speed, flow, density

- **Mesoscopic**
  - Medium level of detail
  - Individual vehicular behavior is not described, but specified by means of probability distribution functions

- **Microscopic**
  - High level of detail
  - Each vehicle considered as a single simulation object
  - Vehicular behavior is acquired from motion of leader vehicle
    - Position, speed, acceleration
  - Computationally expensive
Traffic Models

- **JUTS (Java Urban Traffic Simulator)**
  - 2.5m cells
  - Vehicles can occupy 1-6 cells
  - Vehicles move in cells each step

- **TRANSIMS (Transportation Analysis and Simulation System)**
  - Similar to JUTS with larger cells (7.5m)
  - Each vehicle can occupy only one cell

- **AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks)**
  - No cells
  - Any vehicle length and position is supported
  - 1 second time step
Traffic Network

- Lanes (roads)
- Crossroads
- Curves
  - Merging, dividing of lanes
- Generators
  - Generate vehicles coming into simulated area
- Terminators
  - Remove vehicles leaving simulated area
**Decomposition of Distributed System**

- Cluster of single-process computers (nodes) on a network
- Spatial decomposition of traffic map
  - Traffic spatially divided into sub-networks
  - Vehicles transfer between sub-networks during simulation
    - Synchronization is very important!
    - Message passing communication
**Distributed Execution**

- Each simulation divided into sub-networks, each assigned to a node of the cluster
  - One process per node
- Focus on load balancing for optimal performance
  - Divided by number of lanes, because of vehicle density
- Each sub-network is given a terminator-generator pair in place of straight lane connectors
Distributed Execution: Communication Protocols

- Communication is slower than computation
- Focus on reducing communication
- Semi-Centralized Vehicles (SC-V) protocol
  - Utilizes direct communication between neighboring sub-networks for transfer of vehicles
  - Every step, terminator vehicles are sent to corresponding generator vehicles
  - Centralized process (control process) maintains barriers and synchronization
Semi-Centralized Vehicles (SC-V) Protocol

- **Need for synchronization**
  - Vehicles can travel from one sub-network to another
  - Terminated vehicles in one sub-network must be generated in neighboring sub-network
  - Terminator cells remove vehicle from sub-network 1 and packs it into a message. The message is sent to sub-network 2 and a vehicle with the same data is generated in the generator cell.

- **Synchronous master-slave approach**
  - Slave completes computation and sends notification to master process
  - Upon receiving notifications from all slaves, master broadcasts permission to continue with next time step
  - Either the PVM or MPI interface can be used

- **Reducing communication costs**
  - Send traffic flow data instead of individual vehicles, update only if traffic patterns change
a - Vehicle is removed from the road by the terminator (T)

b - Description of the vehicle is packed into a message and sent to sub-network 2

c - The message is forwarded to the corresponding generator (G) by the message router (MR)

d - The description of the vehicle is taken from the message

e - A new vehicle is generated according to description
**Decomposition of Parallel/Distributed System**

- In this model: cluster of nodes with multi-core processors
- Similar to distributed system
  - Spatial decomposition of traffic map
    - Traffic spatially divided into sub-networks
    - Vehicles transfer between sub-networks during simulation
      - Synchronization is very important!
      - Message passing communication
- Message passing occurs even on processes on the same node
  - Faster, because it’s not over the network
  - Still non-negligible
DECOMPOSITION OF PARALLEL/DISTRIBUTED SYSTEM

- Problem: many more processes → many more sub-networks → more communication
- Solution: instead of assigning one sub-network per process, assign multiple processes (threads) to sub-networks
  - Each process has multiple threads
  - No communication within node due to shared address space
  - Communication only occurs between different nodes
PARALLEL/DISTRIBUTED EXECUTION

- Distributed execution
  - 1 second steps
  - Each step, update (in order) terminators, generators, lanes, curves, crossroads

- Parallel/Distributed (multithreaded) execution
  - Each thread takes portion of lanes, crossroads, curves
    - Dividing job into homogeneous partitions of lanes, crossroads, and curves creates a load imbalance
  - Simply maintaining the same update order is insufficient for synchronization
    - must synchronize after every step using barriers
      - Terminators, barrier, generators, barrier, etc.
TESTING PARAMETERS

- Two test sets
  - Quality of parallelization of threads (on one node only)
  - Comparison of distributed and parallel/distributed parameters

- Parameters
  - Three sets of traffic networks
    - 64, 256, 1024 square grids
    - 86,400 m, 326,400 m, 1,267,200 m total traffic lane lengths

- Each simulation run one hour
RESULTS: QUALITY OF PARALLELIZATION ON THREADS

Dependency of simulation time on the number of utilized cores (threads) for the JUTS-based model

- Using all four processor cores, simulation is on average 3.32 times faster than using only a single core
- Synchronization of threads several times per time-step brings additional overhead to simulation
RESULTS:
COMPARISON OF DISTRIBUTED VS PARALLEL/DISTRIBUTED

- Speedup of the parallel/distributed version of the DUTS system in comparison to the distributed system
  - Speedup from 5% to 52%
- Speedup of the simulation using two working nodes instead of one
  - Speedup from 5% to 47%

- Small speedup for smaller traffic networks because of intense inter-process communication
- Effect of Communication is diminished on larger traffic networks because a larger portion of total simulation time is used on computation
CONCLUSIONS

- Inter-process communication time reduced due to parallelization
  - Utilization of shared address space instead of message passing in between threads
- Parallel/distributed version reaches up to 52% speedup in comparison to distributed version
REFERENCES


