Distributed Systems Project Report

Distributed Simulation

Members:
Arpit Kumar
Sachin Kumar
Siddharth Rakesh
Manav Sethi

Supervisor:
Prof. Arobinda Gupta

April 14, 2015

Contents

1 Introduction

Distributed simulation involves the execution of a single simulation program on a collection of loosely coupled processors (e.g., PCs interconnected by a LAN or WAN). Distributed simulation is used for a variety of reasons, including:

- Enabling the execution of time consuming simulations, that could not otherwise be performed (e.g., simulation of the Internet), which helps in reducing the model execution time (proportional to the number of processors) and provides the ability to run larger models (with more memory).
- Enabling the simulation to be used as a forecasting tool in time critical decision making processes (e.g., air traffic control), wherein the simulation can be initialized to the current system state, and faster than real-time execution can be achieved for what-if experimentation, as the simulation results may be needed in seconds.
- Creating distributed virtual environments, possibly including users at distant geographical locations (e.g., training, entertainment), providing real-time execution capability, scalable performance for many users and simulated entities.

2 Objectives

The objective of this project is to develop a distributed simulation framework for simulating problems such as random walks in a graph. It includes multiple components, which can be represented by the following points:

- Ensuring a near equal workload distribution among multiple nodes, and distributing the graph in a manner, so as to minimize inter-node communications.
- Developing a reliable, FIFO communication framework among the nodes.
- Developing a well-coordinated framework for message handling and processing using multi-threading.
- To design methods for assessing how well the entire system performs.

3 Dataset

- A small synthetic dataset
- A random graph of 100 nodes
- Co-authorship network with 1000 nodes
- Co-authorship network with 100000 nodes

4 Approach

4.1 Distributing the graph

Algorithms that find a good partitioning of highly unstructured graphs are critical for developing efficient solutions for a wide range of problems in many application areas on both serial and parallel computers.

For example, large-scale numerical simulations on parallel computers, such as those based on finite element methods, require the distribution of the finite element mesh to the processors. This distribution must be done so that the number of elements assigned to each processor is the same, and the number of adjacent elements assigned to different processors is minimized.

The goal of the first condition is to balance the computations among the processors. The goal of the second condition is to minimize the communication resulting from the placement of adjacent elements to different processors. Graph partitioning can be used to successfully satisfy these conditions by first modelling the finite element mesh by a graph, and then partitioning it into equal parts.

We have used gpmetis, an open source application which uses multilevel k-way partitioning algorithms to compute a partitioning solution in which each partition is contiguous. The partitioning and ordering routines compute multiple different solutions and select the best as the final solution. Our objective function is Edge-cut minimization. The command-line programs provide full access to the entire set of capabilities provided by gpmetis' API.

4.2 Establishing communication between the nodes

Our framework requires reliable communication because receiving messages correctly is imperative for the proper functioning of the system. FIFO channels are required because each processor maintains a set of event queues for every other machine and a local queue, and we assume that a new message in an event queue has a time stamp, not smaller than the last event.

In order to ensure such reliable, FIFO communication, we use TCP connections between the nodes. Each node in the system acts as a server and client. While acting as the client, it sends connection requests to all other nodes, using their IP addresses stored in a configuration file before the application begins execution. Simultaneously, it acts as a server and accepts connections from the other nodes, which try to connect to it.

Thus, we obtain a fully connected topology wherein each node has a bidirectional connection to every other node via TCP connections.

4.3 Developing the message handling and processing framework

4.4 Designing system assessment methods

To test the system and its capabilities, we have simulated the random walk problem. A random walk is a mathematical formalization of a path that consists of a succession of random steps. For example, the path traced by a molecule as it travels in a liquid or a gas, the search path of a foraging animal, the price of a fluctuating stock and the financial status of a gambler can all be modelled as random walks, although they may not be truly random in reality.

For testing our system, we began by seeding approximately 10 percent of the nodes with random walkers. If any node achieves 250 visits, we terminated the system. At the end, we computed the steady state probability that a node is visited in a random walk using the number of times it was actually visited during the walk.

5 Results Obtained

Based on the tests performed, we computed the Normalized Root-Mean-Square error between the observed and theoretical steady state probabilities. The error obtained is 13.18 percent.

References

[1] Parallel and Distributed Discrete Event Simulation: Algorithms And Applications by Richard M. Fujimoto, Proceedings of the 1993 Winter Simulation Conference
http://dl.acm.org/citation.cfm?id=256596

[2] Distributed Discrete-Event Simulation by Jayadev Misra
http://dl.acm.org/citation.cfm?id=6485

[3] Integrated Fluid and Packet Network Simulations by George F. Riley, Talal M. Jaafar and Richard M. Fujimoto
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1167114

[4] Parallel Simulation of Telecommunication Networks http://titania.ctie.monash.edu.au/pnetsim.html

[5] Introduction to Discrete-Event Simulation and the SimPy Language by Norm Matloff
http://heather.cs.ucdavis.edu/~matloff/156/PLN/DESimIntro.pdf

[6] The OMNET++ Discrete Event Simulation System by Andrs Varga
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.331.1728&rep=rep1&type=pdf

[7] EpiSimdemics: an Efficient Algorithm for Simulating the Spread of Infectious Disease over Large Realistic Social Networks by Christopher L. Barrett, Keith R. Bisset, Stephen G. Eubank, Xizhou Feng, Madhav V. Marathe, Network Dynamics and Simulation Science Laboratory, Virginia Tech, Blacksburg
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5214892

6 Feedback