TRANSIMS-Seminar

Some Aspects of Load Balancing

December 15th, 1994

Marcus Rickert

Why load balancing?

- Because of the simulation (dedicated and non-dedicated): in dependance of the simulation time certain regions of the road network will have fluctuating numbers of objects (rush hours)

- Because of the hardware (non-dedicated):
  - workstations are ususally not fully available to the simulation
  - availability changes due to work pattern of users
  - servers have fluctuations due to IO dependant on the overall load of the system.

The Main Objectives of Load Balancing in a Timestep Driven Simulation

- Equal execution time for the locally residing road network.
- Minimizing the number of neighbours each CPN has (minimizing the number of messages)
- Minimizing the number of boundary edges each CPN has (minimizing the sum length of messages)
- Dedicated systems only: optimizing the mapping of the road network onto the computer network (minimizing the sum distance that messages have to travel assuming that local communication is faster than global)

Implemented Approach

- We assume that each transferable object is able to return a value proportional to its computational requirements in arbitrary units. Together with the actual wall clock execution time the load of a CPN is determined.

- Each CPN computes *one* region of the road network.

- During load balancing road network is only transferred between neighbouring CPN and only along the boundaries so that regions may grow, shrink, and change their shapes, but are *never* split or merged.

- There will be a penalty function evaluating the shape, connectivity and location of the locally residing road network and determining which parts are to be transferred to reduce the penalty (or to have the least increase in penalty)

Initial distribution

Given:

- a road network with nodes and segments, each node $n_1 \dots n_N$ has a geometric location (x,y) and estimated load l_i.

- a CPN network with a relative speed for each CPN $CPN_1 \dots CPN_C$.

Do a recursive split of the networks:

1. If the #CPNs=1, then assign nodes to CPN
2. Split CPNs in halves
3. Sort nodes according to their x(y) coordinates
4. Split nodes in such a way that the ratio of load estimates of both the nodes is equivalent to the ratio of sum speeds of the CPNs
5. Split recursively

After recursion: find unconnected parts of the network and assign them to neighbours.

How to determine the load

- Measure the wall clock time each CPN needs to compute the locally residing road network.
- Measure the "estimated sum load" returned by the objects in the road network.
- Compute the "corrected load" $L := l/t$ and keep a history of the L with a certain history length.
- Take the minimum performance of all values stored in the history and use it for loadbalancing.

A transfers l_t to B:

$$\frac{l_a - l_t}{L_A} = \frac{l_B - l_t}{L_B}$$

$$\longrightarrow l_t = \frac{l_A L_B - l_B L_A}{L_A + L_B}$$

Insertion of a new CPN_N

1. User adds another CPN through PVM interface

2. Master initiates a global synchronization

3. Determine CPN_A that has least idle time

4. CPN_A determines the neighbour CPN_B that has least idle time

5. CPN_A transfers a *single* node that has boundary edge with CPN_B to CPN_N

6. CPN_A informs CPN_B about the transfer

7. Simulation continues

Deletion of CPN_D

1. CPN_D receives a signal SIGHUP
2. CPN_D refuses to accept any road network from its neighbours, but instead tries to offload everything down to a *single* node
3. CPN_D informs the master if node count has reached one
4. The master initiates a global synchronization
5. The master picks out one neighbour CPN_N of CPN_D and prompts CPN_D to transfer the remaining node to CPN_N
6. The master terminates the slave process on CPN_D
7. Simulation continues