CSCI 6360: Parallel Computing Lecture Summary - 10

Anirban Das (dasa2@rpi.edu)

February 16, 2018

Summary on PDES.

The lecture explores Discrete Event Simulators (DES) and their design patterns and the implementation of these DESs on massive parallel supercomputing environment. Many a times we need to simulate real word complex scenarios/ systems for either benchmarking or performance testing or for design changes etc. Discrete Event Simulators are necessary to simulate such large scale dynamical systems such as vehicular networks, power grid, wireless communication, geo distributed data aggregation frameworks, ISP backbones, next gen super computer systems etc .

A DES simulates a series or sequence of discrete events in time where generally one event triggers the next one with no intermediate changes in the system assumed in between. Each event can either modify state variables (which models physical systems) or schedule next event(s) to occur. For e.g. an Air Traffic model would contain state variables such as current simulation time, number of landed aircraft etc and events such as arrival, departure, landed etc. By processing this events chronologically in a chained manner, the main event processing loop, with help of some specific data structures is able to simulate the model.

The problem arises in keeping the event computations synchronised or time stamp ordered in a massively parallel execution. A barriered solution in inherently time consuming and thereby illogical, a non barriered irregular execution has the problem of past events arriving after current events, time-warp method has cascading cancellation/rollback effect and is expensive, null message algorithm can avoid deadlock problem but a livelock can occur. This problem can be solved using a lookahead, which is basically a stochastically generated time interval after which a LP event schedules its next event using logical time. It will process all events withing that interval, and this allows concurrent processing of events, albeit at a performance cost.

Reverse computation paradigm however is a viable solution and is a balance between performance and complexity. The idea is that the compiler automatically generates a reverse code and in case of rollback just executes reverse code to undo. This obviously generates low overhead for forward propagation and since the chances of rollbacks are statistically low, the overheads of reverse code computation and execution causes very little performance issues with decrease in memory usage as an added benefit. The implemented system is known as ROSS and it is installed on an IBM Blue Gene L and P system. MPI_Isend and MPI_IRecv are used to send/recv off core events and the performance is tested on two operations PHOLD, a synthetic benchmark model and PCS, a cell phone call networking model.

The time warp scalability showed that the BG/L system can process upto peak 7.5 billion ev/sec on 32,768 cores on 10% remote and about 2.7 billion ev/sec on 100% remote. The same on a BG/P is a peak 12.27 billion ev/sec at 10% remote for 65,536 cores and 4billion ev/sec for 100% remote, where remote means the amount of randomly routed events among all LPs. The rollback efficiency stays almost stable upto about 64K nodes and then plummets with increasing parallelism.

The result suggests that billions scale ev/sec is achievable in feasible time frame today. This seems fascinating, as part of my research I looked into OMNET++ which is a large scale vehicular

DES with Veins. Normally with this paradigm and scale at disposal, I would imagine it is possible to create similar framework to simulate the behavior of a whole state wide vehicular networking including the overlay networking in a petascale computer only!