RM 스케쥴링과 Lock-Free 공유개체에 의한 실시간 시뮬레이션

박 현 규
전투지휘훈련단
(T) 042-870-2477

The Real-Time Constructive Simulation With the RM Scheduling and Lock-free Shared Objects

Park, Hyun Kyoo
BCTP Corps

Abstract

The Constructive Battle Simulation Model is very important to the recent military training for the substitution of the field training. However, real battlefield systems operate under real-time conditions, they are inherently distributed, concurrent and dynamic. In order to reflect these properties by the computer-based simulation systems which represent real world processes, we have been developing constructive simulation model for several years.

The constructive simulation system is one of the famous real-time system software, and the one common feature of all real-time systems is defined as the correctness of the system depend not only on the logical result of computation, but also on the time at which the results are produced.

Conventionally, scheduling and resource allocation activities which have timing constraints are major problem of real-time computing systems. To overcome these constraints, we elaborated on these issues and developed the simulation system on commercially available hardware and operating system with lock-free resource allocation scheme and rate monotonic scheduling.

1. Introduction

The Army's C2(Command & Control) training simulation models are interactive computer-driven systems that assist training readers, commanders, and their staffs to develop and maintain the unit readiness.

Most work on implementing shared objects in preemptive hard real-time uniprocessor systems has focused on using the priority-driven scheduling and lock-based critical sections to ensure object consistency with a cost ineffectiveness, thus it specifically increased operating system overhead.

In the constructive simulation, lock-free object implementations are of interest because they avoid priority inversion and deadlock with no underlying operating system support for object sharing. Additionally, we can assimilate the user interactions and the inference events as the rate monotonic scheme.

Since, in a hard real-time system, unless priority inversions are carefully controlled, it may be difficult or impossible to ensure that task deadlines are always met.

Theoretical Background And Previous Studies

There are some key definitions that apply through this paper are enlisted below.

Event : A change of object attribute value, an interaction between objects, an instantiation of a

new object, or a deletion of an existing object that is associated with a particular point on the simulation time axis. Each event contains a time stamp indicating when it is said to occur.

Logical Time : A simulation system's current point on the logical time axis used to specify before and after relationships among events. Specifically, the simulation requests advances in logical time via Time Advance Request and the simulation system notify the advance of logical time through the Time Advance Grant service.

Wallclock time : a simulation system's measurement of true global time, where the measurement is typically output from a hardware clock.

Lock-free Object Sharing

The lock-free approach to real-time object sharing that we investigated was done by previously some twenty years ago in the real-time systems community. But this idea was forgotten for many years.

Representative lock-free operations are usually implemented using "retry loops," such as figure-1. It depicts a lock-free operation example which is implemented in this way. In figure-1, a message is inserted in the buffer by using a *write* instruction to update the buffer status by changing a tail pointer and either the next pointer of the last item in the buffer or a head pointer, depending on whether the buffer is empty. This loop is executed repeatedly until the *write* instruction succeeds.

The buffer is not explicitly locked by any task, so it is essential property of lock-free implementations that operations may interfere with each other. An interference results in this example when a successful *write* by on task results in a failed *write* by another task. However, it is not immediately apparent that lock-free shared objects can be employed of tasks must adhere to strict timing constraints.

```
class SharedBuffer
public variable
    Head, Tail : pointer
    AccessKey : Integer
method write (stream)
private variable
    Old, New : pointer
    do
        if Old != Null then add stream;
        else add stream; send signal;
    until EventRemaind()
```

Figure-1 : Lock-free Shared Buffer Implementation

Rate Monotonic Real-Time Scheduling

Scheduling real-time computations is an extremely important part of a real-time system because it is the phase in which we assign the actual temporal properties to the computations For many years, Rate Monotonic(RM) scheduling theory offers a set of engineering principles for managing timing complexity.

To apply RM scheme in this simulation, it is assumed that task deadlines and periods coincide, i.e, each invocation of a task must complete execution before the next invocation that task begins.

Under RM scheme, the necessary and sufficient conditions for the schedulability of a set of periodic tasks that share lock-free objects were formerly investigated. For brevity, we omit the proof of this condition here, and the formal proof within our model can be found in [3].

The following two theorems give the necessary and sufficient conditions for the scheduling by RM scheme in this simulation.

Theorem 1. (Necessity under RM) If set of periodic tasks that share lock-free objects is schedulable under the RM scheme, then the following condition holds for every task T_i.

$$\exists t : 0 < t \leq p_i : \sum_{j=1}^{i} \lceil \frac{t}{pj} \rceil \cdot c_j \leq t$$

The left-hand side of the quantified expression gives the minimum demand - which arises when there are no interferences - placed on the processor by T_i and higher-priority tasks in the interval [0, t] where $0 < t \leq p_i$. The right-hand side gives the available processor time in that interval.

Theorem 2. (Sufficiency under RM) A set of periodic tasks that share lock-free objects is schedulable under the RM scheme if the following condition holds for every task T_i.

$$\exists t : 0 < t \leq p_i : \sum_{j=1}^{i} \lceil \frac{t}{pj} \rceil \cdot c_j + \sum_{j=1}^{i-1} \lceil \frac{t-1}{pj} \rceil \cdot s \leq t$$

where

p_i : The period of task T_i, $p_i < p_j \Rightarrow i < j$

T_i : i_{th} task in the system

c_i : The worst-case computational cost of task T_i

s : The execution time required for one loop iteration in the implementation of a lock-free object, which for simplicity is assumed to be the same for all objects

Time Stamp Event Ordering

We have limit timing complexity with following constraints. The time point is a real number that represents the occurrence time of an instantaneous event, and is an individual entity.

Each event is able to access a clock and acquire the estimate of the current logical time $F_i(t)=t \pm \Delta i(t)$, through the time service. $F_i(t)$ is the monotonic function that maps real-time to logical time and Δi is the latency by the network and operating system overhead.

Events are delivered to Event-Handler in time stamp order. The events will not be processed until the event with a smaller time stamp being remained. To accomplish this task, Event-Handler will hold incoming events in its queue by the time stamp and also ensure that no event is delivered to a Event-Handler "in its past," i.e., no event is delivered that contains a time stamp less than the system's current logical time. This eliminates certain temporal anomalies that might otherwise occur when various clients send different ordering events. And each event has only convex(contiguous) duration until completion of execution in this simulation.

Major Features of Implemented Model and Experimental Results

The simulation model is constructed on single CPU Sun Ultra-workstation and IBM PCs connected via local network for clients. The clients have their own Geographical Information System(GIS) module and user interaction components to transmit the events and receive the result messages.

For the logical time in this simulation system, we use the Unix software clock(Solaris clock) and distribute the logical time to all processes to synchronize the entire system. This time service called Time Advance Grant(TAG) distribute the logical time each minute by the progress ratio. Also, sometimes the progress ratio would be changed by the client request and then this task will update the scale factor of software clock either gathering the logical time or loosing time through Time Advance Request (TAR).

Each event has time stamp by the logical time and ordered by its time stamp before input to the event queue.

Lock-free scheme as described, the shared memory is widely used for the message transmission among the processes and client nodes which are attached via the network and controlled by *network daemon*.

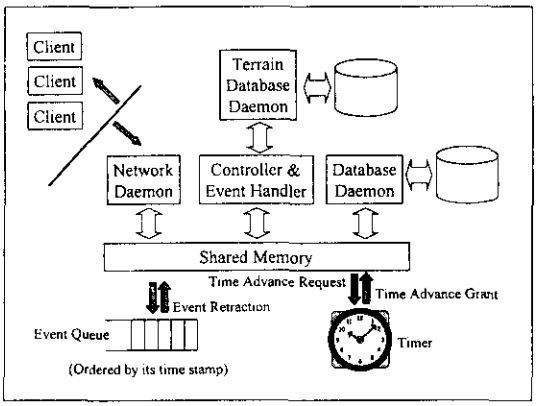

Figure-2 : Simulation Model Architecture

The event-handler has the key role of simulation both for the logical computational result and the control of event processing (ordering, retraction, etc). Through the experiments, this simulation model supports our requirements with previously described ideas successfully.

But, in this paper, we omit the data gathering and report generating system which is important part of the client operation. For the attribute of simulation system, this model is consisted with a separate data gathering and reporting communication channel, which trigger the database engine - especially Informix*. This means that the simulation system has unpredictable long-term events. This infrastructure constitutes the basis for aggregation of simulation results and reporting of the running system under real application load.

Guaranteed worst-case response

One of the most distinguishing feature of the static priority scheduling is that it can't provide guaranteed timely services to real-time applications when the importance of task is changed. The dynamic or adaptive scheduling might resolve this problem.

By the constraint of event raising intervals and durations, we can adhere the static priority with lock-free object sharing through the adaptation of logical time service. The Time Advance Request will change in the time progress ratio adaptively when the event execution is not expected to meet the deadline.

Concluding Remarks & Future Works

The time management structure is intended to support interoperability among heterogeneous simulation systems utilizing different internal time management mechanisms. Eventually this execution support distributed time advance mechanism for the next generation distributed simulation model [5]. And our preliminary results proved that the lock-free shared object with static priority scheduling is affordable for the real-time simulation on uniprocessor system. These efforts will remove special hardware and the problem-oriented real-time operating system support for the wargame area. However, this scheme needs further experiment to decide if it can be effectively applied to the hard real time simulation.

참고문헌

[1] Levi and Agrawala, " Real-time System Design", McGrow-Hill, 1990.

[2] Training with Simulations, National Simulation Center, Nov. 1996.

[3] James H. Anderson, et. al, "Real-Time Computing with Lock-Free Shared Objects", Proceedings of the 16th IEEE Real-Time Systems Symposium, Dec. 1995.

[4] 박현규, "Unix 다중프로세서 시스템의 성능 측정", 국방정보관리, 1997.

[5] 박현규 외, "TMO모델을 이용한 분산 실시간 워게임 시뮬레이션 모델의 개발", 시뮬레이션 학회, 1998.