STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XL, 3, 1995

AN EVENT-DRIVEN SEQUENTIAL SIMULATION ALGORITHM FOR DISTRIBUTED SYSTEMS PERFORMANCE EVALUATION

Alexandru VANCEA* and Monica VANCEA**

Received: November 15, 1995
AMS subject classification: 65Y05, 68M20

REZUMAT. - Un algoritm de simulare secvenţială orientată pe evenimente pentru evaluarea performanţei sistemelor distribuite. Determinarea performanţelor de execuţie ale sistemelor distribuite reprezintă o activitate de cercetare în continuă dezvoltare, atât în ceea ce priveşte criteriile de performanţă principale care ar trebui adoptate cât şi natura algoritmilor care trebuie utilizaţi într-un astfel de studiu. În lucrare se susţine efectuarea de analize de performanţă prin aplicarea unor algoritmi care să simuleze secvenţial execuţia aplicaţiilor care se rulează într-un sistem distribuit. Lucrarea prezintă un astfel de algoritm orientat pe evenimente (event-driven) care funcţionează într-un mediu distribuit în care comunicarea se face prin transmitere de mesaje.

1. Preliminaries. The actual development of distributed software systems creates new requirements for both programming languages and operating systems. Even if distributed computing had become a very active research area in the last few years, the complexity imposed by the probability features involved in the functioning of a distributed computing system makes it very difficult to accurately determine the execution performances through a simple (static or dynamic) analysis. We find many situations in which one can derive only some lower bounds and upper bounds with a large difference between them, making (almost) meaningless the whole study.

Taking into account the inherent difficulties posed by a real-time study of a distributed execution with the actual hardware and software possibilities, we claim in this paper that sequential simulation of a distributed execution is still an important and effective method for

* *"Babeş-Bolyai" University, Faculty of Mathematics and Computer Science, 3400 Cluj-Napoca, Romania*

** *"Babeş-Bolyai" University, Faculty of Economics, 3400 Cluj Napoca, Romania*

evaluating the performance of a distributed system.

It is very hard to find simulation models for distributed systems. There are many studies involving queue's models, but it seems finally not to be a very good choice because in distributed systems is little central control and generally the nodes have equal importance.

Sequential discrete event simulation can be made in two forms: *event driven* and *time-driven*. In the event-driven simulation events which appear in the physical system are simulated chronologically and the simulation clock is advanced after the simulation of an event to the time of the next event to be executed. In the time-driven simulation the clock advances by one tick in every step and all events scheduled at that moment in time are simulated.

The simulation considers buffered communication and takes into account only communications between adjacent nodes made through message passing. The communication between non-adjacent nodes can be transformed into communication between some pairs of adjacent nodes by the distributed path choice algorithm [Duan87].

We start our discussion from the method presented in [Ping87] which implements for every node a so called *synchronization management module* (SMM). Every module exchanges its status information by reading and writing some boolean flag variables in order to implement the synchronization of interprocessor communications. The SMMs of the different nodes have the same structure, the only difference between them being the values of some parameters which depend on the node information. In each time phase of an operation each module will perform one of the following actions:

(1) *Asking phase*: try to send a message to an adjacent node;

(2) *Responding phase*: accept messages from adjacent nodes;

(3) *Closing phase*: neither send to nor accept message from an adjacent node;

The choice is made accordingly to the values of two status variables (*buffer_full* and

send) which indicate respectively whether the node's communication buffer is full or not and whether the current node needs or not to send a message to another node during the current fraction of time.

During distributed systems simulations the following assumptions are frequently involved [Lan84]:

(1) When a message is to be sent, the probability to be the sender node is equal for every node; also is equal the probability of every other node to be the target node;

(2) The frequency at which a node sends messages follows a negative exponent dostribution;

We mentioned that we consider only communications between adjacent nodes, so assumption (1) is changing in:

(1a). When a message is to be sent, the probability to be the sender node is equal for every node; also is equal the probability of every one of its adjacent nodes to be the target node;

2. Event-driven sequential simulation. The basic intuitive algorithm for the event-driven sequential simulation is constructed as following:

a). Input the interconnection structure of the distributed system for which the algorithm is used; set the initial status and time for each node;

b). Select the node for which the current time is the smallest;

c). Simulate the activity of this node for one or more phases and update the data structure;

d). If the number of the sent messages is not adequate go to (b), otherwise output the simulation result;

The simulation is based on the assumptions mentioned above. Two kinds of stochastic

variables are involved in the simulation system: one of them follows uniform distribution and the other follows negative exponent distribution. The function F(i+1) = (a*F(i)+b) mod m is used to generate a pseudo stochastic number, function for which the following theorem is known [Lan84]:

Theorem 1. The period of the pseudo stochastic number generator F(i+1)= (a*F(i)+b) mod m is m, if and only if

i). b mod m = 0 and m mod b = 0;

ii). for any prime number p such that m mod p = 0, we have

(a-1) mod p = 0;

iii). if m mod 4 = 0 then (a-1) mod 4 = 0;

In general, $m = 2^k$, where k is a positive integer.

The function δ = max {F(n), 1}/m is used to generate the stochastic real number which follows a uniform distribution in the (0,1) interval. From this, we obtain:

a). Negative exponent distribution with mathematical expected value $1/\lambda$: $\eta = -(\ln \delta)/\lambda$;

b). Uniform distribution in the set {1,2,...,k}: $l = \lfloor k*\delta \rfloor$

Let λ_i be the joint probability of the following probabilities:

(1) the probability of node i to send a message to another node per phase;

(2) the probability that the communication buffer of node *i* to be full;

Let N be the number of nodes and let $R = \lfloor R_{max}/R_{min} \rfloor$, where R_{max} and R_{min} indicate the upper and lower bounds respectively of the node's speed, $0 < R_{min} \le R_{max}$.

In order to simplify the simulation process we can assume that:

(a) the ratio of node i's speed to R_{min} is 1+i*(R-1)/N, and it does not change in the running process;

(b) all λ_i have the same value, where $1 \le i \le N$;

(c) the communication buffer is large enough for never be completely full, so, no closing phases will be encountered during the simulation process;

These assumptions have no inherent effects on the validity of the simulation results.

Because the nodes speeds are different, the nodes upper bounds of one phase's length will be different. Let $upp_{l(i)}$ be node's i upper bound of one phase's length and status(S) and ctime(S) be the current status and the current time respectively of node S. By n(S) we denote the number of phases for which node S has successively requested to communicate.

With these considerations the general form of the simulation algorithm is as follows:

```
begin
      initialize the interconnection structure of the distributed
      system, R, λ and other functional parameters;

      while (the number of sent messages is not enough) do
            select a node S with min(ctime(S));

            case status(S)=0 :

      { 1 - simulate the environment in which the algorithm works
      according to the negative exponent distribution, compute the
      length t of the interval between ctime(S) and the moment at which
      node S wants to send the next message; according to the uniform
      distribution select the target node;}

                  ctime(S)=max(ctime(S), ctime(S)+t);

      { where t is the response time of node S's last communication }
      { 2- record related information }

                  status(S):=1;

            case status(S)=1 :  { simulate the responding phase }

                  n(S):=0; status(S):=2;

            case status(S)=2 :  { simulate the asking phase }

                  if n(S)=S then status(S):=1 else
```

```
                    begin
                     if (the communication request has been
                                              succesfully answered) then
                        begin
                          compute and record related information;
                          status(S):=0;
                        end
                     else n(S):=n(S)+1;
                    end;

                  if status(S)=0 then
                   ctime(S):= the time of node S at which the
                               communication is established
                  else
                   ctime(S):=ctime(S)+upp_{l(S)};
            endcase;
      endwhile;
      output the simulation results;
end.
```

The performances and precision of such an algorithm are now under practical research on the Sun network stations at UBB Computing Centre and the results will be reported comparatively to some other related algorithms in a future paper.

REFERENCES

[Akl90] S.G.Akl - *The Design and Analysis of Parallel Algorithms*, Prentice Hall, 1989.

[Duan87] Ping Duan and Cai Xiyao - A Real-time Interprocessor Synchronization Algorithm for Communications in Distributed Computer Systems, in *Journal of Computer Science and Technology*, vol.2 (1987), no.4, pp.292-302.

[Krish89] E.V.Krishnamurthy - *Parallel Processing. Principles and Practice*, Addison-Wesley, 1989.

[Lan84] Jin Lan and Zheng Weimin - Stochastic Simulation of a Distributed Multiprocessor System, in *Chinese Journal of Computers*, vol.7 (1984), no.2, pp.100-107.

[Ping87] Duan Ping - A Real-Time Synchronization Algorithm for Interprocessor Communications Based on Duplex Systems, in *Proceedings of IEEE Asian Electronics Conference*, 1987, pp.473-475.

[Tan92] A.S.Tanenbaum - *Modern Operating Systems*, Prentice Hall, 1992.