Asian Journal of Information Technology 6 (3): 309-313, 2007

Parallel Simulation of Manufacturing Systems

[1]Abdelhak Boubetra, [1]Nacereddine Mouhoub and [2]Hocine Belouadah
[1]University of Bordj Bou Arreridj, Algeria
[2]University of Msila, Algeria

Abstract: Given the widespread of manufacturing systems, it is important to understand their behavior through simulation. The proposed simulation approach views this type of systems through an analogy with 'the producer-consumer' standard computer problem and its advantage to enable computer parallelism to be achieved in order to optimize computationnel time to simulate such systems.

Key words: Manufacturing system, component, optimization, event, simulation, parallelism

INTRODUCTION

Manufacturing system simulation utilises computational models that allow the user to study manufacturing processes without the requirement to invest in a real implementation. These systems are composed of objects or entities (e.g., machines) having attributes or properties. The values of the attributes give the state of an entity and the states of all the entities give the state of the system. The states change due to activities of the entities of the system.

In manufacturing systems, changes of state occur at discrete times and these are called events. Such systems are usually studied at these times only. Hence the manufacturing system is analyzed and studied in terms of events and actions at that events. These last can be classified either time-driven 'occur at some known time in the future' or conditional 'occur when some conditions are satisfied' immediately after a time- driven event.

In the phase of running the simulation, a simulation executive is needed to manage, schedule actions and execute them as they should be executed in the real world. At this stage, we know that simulation is a computer time consuming and it is interesting to have an optimized manner to run the simulation in terms of computer operations time. To deal with that, many time-parallel simulation methods have been suggested for developing massively parallel algorithms for specific simulation problems than a general approach for executing arbitrary discrete event simulation.

A discrete event simulation model of a manufacturing system can be usually considered as a number of connected components (e.g., machines connected by part routes) that receive and consume something in the input to produce something in the output.

Hence, we investigate the simulation of such systems through an analogy with 'the producer-consumer' computer problem.

The study is organized as follows: First three is an introduction to manufacturing systems and the principles of event simulation. Also there is an introduction to parallel discrete event simulation. Then, we emphasis the analogy between the 'producer-consumer' problem and the behavior of a manufacturing system component (e.g., one machine) and we generalize that to networked components (several connected machines). This analogy is explored to show how a simulation program can be transformed into a set of parallel processes reflecting the same simulation model of the system under study. Algorithms specifications of such an analogy are presented.

PARALLEL DISCRETE EVENT SIMULATION

Parallel Discrete Event Simulation (PDES) referes to technologies that enable a simulation program to execute on a computing system that containing multiple processors. Zeigler (1991) emphasised that parallel computing systems are essential for the simulation-based design of complex computer-based systems including computer architectures. Fujimoto sees the principal benefit from executing a simulation across multiple computers is in the reduced execution time gained by subdividing a large simulation computation into many sub-computations and executing the sub-computations concurrently. In other words, PDES techniques are used to speed up the execution of a simulation. Several PDES schemes (Ayani, 1993; Madisetti and Hardaker, 1993) e.g., have been proposed in the recent years with a study of their performance. For Fujimoto and Hybinette (2001)

Corresponding Author: Abdelhak Boubetra, University of Bordj Bou Arreridj, Algeria

the parallel simulator consists of distinct components called Logical Processes (LPs). Each LP immitates and models some behavior of the system under investigation. The logical processes are executed on different processors and interact via messages invoked by events. The simulation advances as LPs exchange messages that model events at discrete points in time. In a special issue on parallel discrete event simulation, Lin (1993) dressed many cases in which PDES is desired.

SIMULATION OF A SIMPLE MANUFACTURING SYSTEM

Consider a system comprising many identical machines which receive and process jobs (Fig. 1).

These jobs wait in a queue before processing. They are treated following a certain discipline.

In this system, the entities are jobs and machines having the following attributes:

Job attributes:

- Waiting to be processed,
- Being processed,
- End of processing.

Machine attributes:

- Busy,
- Idle.

When modeling this system we distiguish the following events:

- Arrival_of_jobs: A time-driven event.
- Start_processing_job: A conditional event.
- End_of_processing_job: A time-driven event.

In a discrete event simulation, the computational actions of these events are:

Arrival_of_jobs:

- Calculate time of arrival,
- Create job record,
- Insert record in queue.

Start_ processing_ job:

- Select job and machine,
- Calculate time of end processing,
- Remove job record from queue.

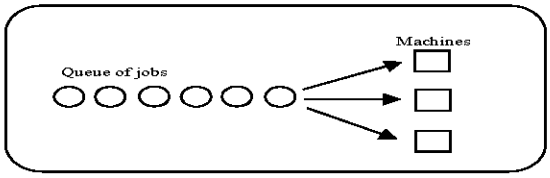

Fig. 1: A simple manufacturing system

End_of_processing_job:

- Annonce end of job,
- Free machine.

On the other hand, when running simulation we are interested in collecting data that serves as statistics to understand the system behavior. In our example these statistics can concern:

- The job information the time of arrival and the time of start processing determine the queuing time of a job and the time of start processing and the time of end processing determine the processing time of a job.
- The machine utilisation is determined by busy time or idle time either at times of start processing and end processing during simulation or by analysis of jobs records.
- The queue size usually obtained by regular sampling of queue at regular time intervals.

The computational actions of collecting statistics data can be defined in a time-dirven event (Statistics_event).

As we know, in the heart of any discrete simulation system there is a calender containing a list of records of time-driven events ordered on time of events and the structure of the simulation program has the following form:

```
Initialise;
While simulation continues do
    Select earliest event from event list;
    Perform actions of that event;
Enddo
Analyse and print results;
```

THE PRODUCER-CONSUMER PROBLEM

In the producer-consumer standard parallel computing problem there are two processes that work in parallel as follows as given in Fig. 2.

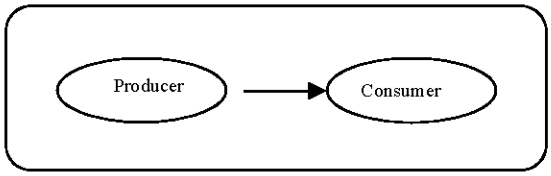

Fig. 2: The producer-consumer problem

Consumer: Removes jobs from queue and processes

Producer: Produces jobs which are queued

An analogy of the producer-consumer problem can be applied to the manufacturing system of Fig. 1 as illustrated in Fig. 2. Hence, the producer and consumer processes will execute the following computational actions as it follows:

Producer:
- Creates arriving jobs,
- Places jobs records on queue.

Consumer:
- Selects job and machine for processing,
- Removes job record from queue,
- Arranges end of processing of job.

With this parrallel computational configuration we can notice that the producer and consumer share the following data:

- Queue of waiting jobs,
- The size of the queue (Q),
- The time of most recent arrival.

Beside that, the queue of the waiting jobs is ordered in time of arrival (Tarr) where :

$Tarr_1 = Tarr_2 = Tarr_3 = \ldots\ldots\ldots\ldots\ldots\ldots = Tarr_n$ (Fig. 3)

This queue is examined by the consumer at time (Tnow). This examination takes the following form:

If ($Tnow > Tarr_n$) {time of most recent arrival } then
    There may be additional jobs arriving until time Tnow is reached
So Wait
Or

If ($Tnow = Tarr_n$) then
    Queue consists of all possible jobs which arrived before Tnow.

A job r belongs to Queue of waiting jobs at time Tnow if $Tarr_r = Tnow$.

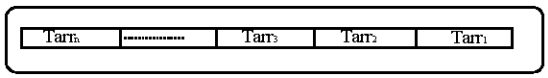

Fig. 3: Queue of waiting jobs

Hence:
- We insert jobs into queue according to time of arrival.
- We select job according to queue discipline.

PARALLEL SIMULATION OF A SIMPLE MANUFACTURING SYSTEM

To simulate the manifacturing system of Fig. 1, we suggest three parallel processes:

Producer procees:
Generates arrivals (arrival events)

Consumer process:
Start processing and end processing of jobs (start and end processing events)

Statistics process:
Using records of processed jobs to obtain job statistics and machine statistics (statistics events).

Here we notice that the queues and shared variables are the waiting jobs with q1 size, the processed jobs with q 2 size and the time of most recent arrival.

The simulation program is written with three parallel processes :producer, consumer and statistics permitting any number of machines, any job discipline and any machine discipline It takes the following structure:

REPEAT
    Producer;
    Consumer;
    Statistics;
    UNTIL simulation finishes

The simulation is terminated when a given number of jobs had been processed or a given time period is elapsed.

The actions executed by the three parallel processes are expressed in the following algorithms:

Producer process
    If q < qmax then
        Calculate time of arrival;
        Create job record;
        Place record at tail of queue;

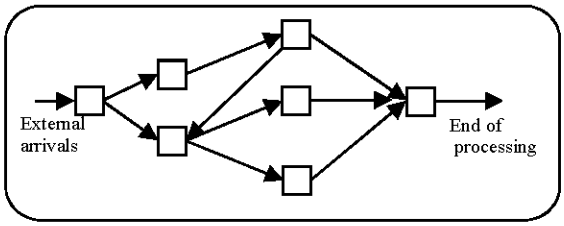

Fig. 4: A complex manufacturing system

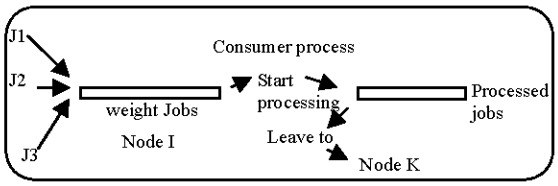

Fig. 5: A manufacturing system node

```
    Increment q;
        End
        Else do nothing { wait }
Consumer process
        {essentially event-driven simulation }
        if q > 0 then
        if number of idle machines = 1 then
            find earliest of arrival, endofprocessing
    else
            find earliest endofprocessing
            (time isTnow)
    end
else do nothing {wait}

arrival:
    select an idle machine;
    start processing with job, machine at time Tnow{time
    of arrival}

Endofprocessing:
    machine becomes idle;
    if number of idle machines = 1 and Tnow = time of
    most recent arrival then
    search queue to select job;
    start processing with job, machine at time Tnow
    {time of endofprocessing}

Start processing {given job, machine, start time}
    Machine becomes busy;
    Insert data in job record;
    Place record in processed jobs queue;
    Set up endofprocessing event;
```

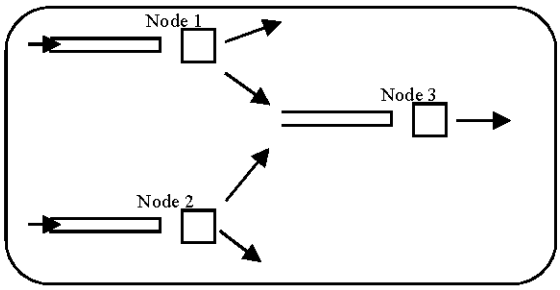

Fig. 6: A manufacturing system example

```
Statistic process

If q processed jobs > 0 then
    Fetch job record from queue;
    Calculate statistics on job, machine;
Elsedo nothing {wait }
```

PARALLEL SIMULATION OF A COMPLEX MANUFACTURING SYSTEM

A manufacturing system can be seen as a network consisting of a number of nodes (Fig. 4) where each node is a set of machines and a queue of jobs that move around network. Jobs can be external arrivals and end of processing.

If we apply the above introduced concepts of the producer-consumer problem to simulate this system we will have the following parallel simulation model:

Producer process(es):

- To generate external arrivals

At each node I:

- Consumer process start processing and end of processing of jobs
- The queue of waiting jobs is organized in order of time of arrival with q size.
- Times of most recent arrivals from all arrival sources for this node {other nodes j or external }.

Let's take a given node I and analyze it from the producer consumer point of view like shown in Fig. 5:

At node I jobs could arrive from nodes J1, J2, J3 and every departure time from these nodes when job comes to node I is a time of most recent arrival to node I from that node. Jobs completing processing at node I may go to

another node K for further processing. Here in the consumer process a job record is removed from queue of waiting jobs at start processing then a job record is transferred to waiting queue of node K at time of departure or inserted into processed jobs list if end of processing of it.

For example, consider a manufacturing system (Fig. 6) consisting of three nodes (node1, node2 and node3) where there are external jobs arrivals at nodes 1,2 and jobs departing from nodes 1,2 either go to node 3 or quit the system.

We notice here that at each node we can have any number of machines and any job or machine discipline. On the principles described above the simulation model will be consisted of the following processes:

Producer 1: To generate external arrivals 1

Producer 2: To generate external arrivals 2

Consumer 1: Start processing and end processing jobs at node1

Consumer 2: Start processing and end processing jobs at node2

Consumer 3: Start processing and end processing jobs at node3

CONCLUSION

The concepts reported in this paper demonstrate a possible PDES of manufacturing systems based on theproducer consumer computing problem. The proposed approach provides a powerful basis destinated to design and simulate a manufacturing system modeled in terms of connected nodes that exchange jobs. This advantage can be extended to Flexible Manufacturing Systems (FMS) because the FMS are being redesigned and reconfigured frequently and rapidly and their performance should frequently be simulated.

REFERENCES

Ayani, R., 1993. Parallel discrete event simulation on SIMD computer, J. Parallel and Distributed Computing 18: 501-508.

Hybinette, M., R.M. Fujimoto, 2001. ACM Transactions on modeling and computer simulation, 11: 378-407.

Lin, Y.B., 1993. Special issue on parallel discrete event simulation, J. Parallel and Distributed Computing 18: 391-394.

Madisetti, V.K. and D.A. Hardaker, 1993. The MIMDIX environment for parallel simulation, J. Parallel and Distributed Computing, 18: 473-483.

Zeigler, B.P., 1991, Object-oriented modeling and discrete-event simulation, Adv. Computers J., 33: 67-114.