Increasing the Efficiency of Construction Simulation Modelling by using Parallel Processing

I Flood and R Pilcher

Department of Building Engineering
University of Manchester
Institute of Science and Technology
PO Box 88
Manchester M60 1QD, England

KEYWORDS

Construction Simulation, Multiprocessor, Parallel Processing, Simulator.

ABSTRACT

The paper discusses the development and evaluation of a parallel computer architecture dedicated to simulation modelling of construction activity, arising out of research into more efficient methods of using computer-based simulation techniques.

A brief description is given of parallel computers, with particular reference to multiprocessors. Previous research is reviewed briefly, discussing the ways in which multiprocessors can be applied to simulation and the advantages which may be gained over the use of conventional computing techniques. The special points to be considered in optimizing the efficiency of a simulation implementation on this type of computing device are discussed.

A diagrammatic method of representing construction systems as a basis for creating simulation models, is introduced. The forms of parallelism apparent in the models produced using this method are assessed and, that form which results in the most efficient implementation on a multiprocessor is established. From this, a multiprocessor-based simulator for modelling construction activity is proposed. Part of this proposal describes a novel approach for both keeping account of simulation time and ensuring that events occur in their correct order during the course of a simulation.

A prototype of this simulator is evaluated by comparing its performance over a range of modelling situations with both that of a micro-type computer and a powerful mini computer. A general discussion of these results is presented.

The paper concludes with suggestions for a number of areas for future research.

Comment Augmenter l'Efficacité de la Modélisation de la Simulation de Construction en utilisant le Traitement en Simultanéité

I Flood et R Pilcher

Department of Building Engineering
University of Manchester
Institute of Science and Technology
PO Box 88
Manchester M60 1QD, England

MOTS CLÉS

Simulation de Construction, Multicalculateur, Traitement en Simultanéité, Simulateur.

ABSTRACT

Nous discutons dans notre article du développement et de l'évaluation de la mise en place d'un ordinateur simultané avec pour but la modélisation de simulation de l'activité de construction, ceci à la suite de recherches sur des méthodes d'utilisation de techniques de simulation basées sur ordinateurs plus efficaces.

Nous donnons une brève description des ordinateurs simultanés, en nous référant plus particulièrement aux multicalculateurs. Nous décrivons brièvement les travaux de recherche précédents, y compris la façon dont on peut appliquer les multicalculateurs à la simulation et les avantages de ceux ci par rapport à des méthodes conventionnelles. Nous discutons ensuite des problèmes spécifiques qu'il faut prendre en compte si l'on désire maximiser l'efficacité de la mise en oeuvre de la simulation dans ce type de système.

Nous introduisons ensuite une méthode schématique de représentation des systèmes de construction, en tant que base pour la création de modèles de simulation. Nous évaluons les formes de parallélisme apparent dans les modèles produits en utilisant cette méthode, et décidons de la forme qui donnera la mise en oeuvre la plus efficace sur un multicalculateur. A partir de là, un simulateur basé sur multicalculateur est proposé pour une activité de construction de modélisation. Une partie de cette étude décrit une aproche nouvelle, tant pour surveiller le temps de simulation, que pour s'assurer que l'ordre de déroulement des diverses phases est respecté au cours de la simulation.

Nous évaluons un prototype de ce simulateur en comparant sa performance dans une variété de situations de modélisation avec celle d'un micro ordinateur et celle d'un mini ordinateur puissant. Nous présentons ensuite des commentaires généraux sur ces résultats.

La conclusion de l'article suggère bon nombre de domaines vers lesquels des recherches vont s'orienter à l'avenir.

1. Introduction

Computer-based discrete-event simulation is potentially a powerful modelling tool for use by construction managers in assisting with the design, estimation, planning, programming and control of all types of construction activity [1]. It is an effective means of assessing possible problems and inefficiencies in a construction process, and their causes, as well as for predicting and optimizing project duration, costs and productivity. Despite this potential, simulation has tended to be neglected within the construction industry due to the comparative expense in its use of both human and computer resources.

The high cost in human resources has been most apparent in the model development stage of the simulation process. Often, the development and commissioning of a model would require several months work from a team of two or three computer scientists/ systems analysts. This problem need no longer prevail. With the development of increasingly sophisticated software packages (in the form of computer-based simulation modelling languages designed specifically for the construction situation [2,3], it is now possible for construction managers with little or no computer expertise to develop models of complex construction systems in a matter of hours.

However, the high demand on computer hardware resources is a problem that has persisted. Even simple simulation models involve extensive computation and thus tend to involve lengthy execution despite continuing improvements in computer technology. Consequently, it is often the case that simulation results cannot be produced quickly enough to be of practical benefit to construction managers, particularly when the model being run is large and detailed. The time required to process a very complex model is likely to be so long that final results may take several days to produce. Even if such processing periods are acceptable, interactive simulation experimentation with such a model will certainly be precluded by tedious delays between the issue of interactive commands by a user and the feedback of information. The use of a large powerful computing system to reduce the time of processing a simulation is not always a satisfactory solution to this problem. A large computing system (as well as being expensive to use) will normally be shared by several concurrent users, and thus will be prone to significant degradation in performance.

A computing device dedicated to construction simulation has been developed in the Department of Building Engineering at UMIST, Manchester to overcome the problem of lengthy processing. This simulator (the development and evaluation of which is the subject of the paper) was designed specifically to run models built using the ICONS [2] simulation language.

2. Parallel processing

Typically, discrete-event simulation models are implemented as a serial algorithm on a general purpose digital computer, whereby each computing operation in a simulation run is executed in sequence by the computer. An alternative computing technique not in common usage, but which can increase greatly the speed at which a simulation is executed, is parallel processing. This requires the use of a parallel computer - a specialized computing device consisting of a number of digital processing elements which can operate

simultaneously, each on its own section of the overall computing task.

There are many different types of parallel computer (a discussion of which is given by Hockney and Jesshope [4]), though a simple means of classification [5] is based on whether or not all processing elements must first, execute the same sequence of instructions and secondly, operate on the same sequence of data. The multiprocessor is an example of a parallel computer where all processing elements can operate independently of each other in both of these respects. In effect, each processing element can follow its own program. This facility makes the multiprocessor highly flexible in programming and a viable medium for applying parallel processing to simulation. Fig 1 shows an outline of a multiprocessor where each processing element has access to both its own private memory and a shared memory. Each private memory may contain program code and/or data specific to its processing element. The shared memory, on the other hand, may contain program code and/or data common to several processing elements and, can be used as an area by which processing elements communicate information.

Korn [6], Pimentel [7] and Nakagawa et al [8], amongst others, propound the use of a multiprocessor to increase the efficiency with which a simulation is executed. Efficiency as such (measured as the ratio of processing speed to hardware cost) is usually cited as the primary advantage of multiprocessing. However, Dekker et al [9, 10] emphasize that a parallel computing device can be used to enhance significantly the extent to which a user can interact with a simulation. To achieve this, they note that the implementation should be made in such a way that there is a one-to-one correspondence between the structure of the computer and that of the system being modelled. Pimentel [7] adopted this philosophy in his multiprocessor-based simulation of an internal combustion engine. He noted that by partitioning the object system into its physical components and allocating each of these to its own processing element, the simulation processes occurring within the computer are easier to interpret and the development of the software is simplified. Alternatively, a simulation program can, for example, be split into a number of functional tasks (such as activity duration generation, event checking and simulation time advancement) each of which can then be allocated to its own processing element. Such an approach, however, is unlikely to procure any benefits other than fast processing.

In spite of these advantages, programming a multiprocessor requires specialized computing skills. Unless special attention is paid to the way in which a simulation is implemented on a multiprocessor, it is unlikely either that a significant gain in processing efficiency will be realized, or that the simulation programs will function correctly. Key factors in achieving an efficient implementation on a multiprocessor are:

a) The number of tasks in parallel execution should be maximized at all stages in a simulation.
b) All processing elements should have a balanced workload throughout a simulation run, since a simulation cannot progress any faster than the processing element with the most work to perform. An exception to this rule can be when a multiprocessor is built from different types of processing element. In this case each processing element can be apportioned a workload corresponding to its performance characteristics.

c) Computational overheads resulting from, for example, the communication of information between processing elements and the sychronization of processing elements at crucial points in a simulation, should be minimized.

The extra skills required for programming multiprocessors is one of the major reasons why this type of device has not been adopted widely within the field of simulation. The aim of the work described in this paper, however, was to develop a multiprocessor-based simulator specifically for the ICONS construction simulation package. In this way, a user of the system can be provided with all the benefits of parallel processing without having to become involved in programming the device.

3. The ICONS simulation language

A detailed description of the ICONS simulation language is presented in the Users' Manual [2]. However, for the purpose of this paper, an understanding of the ICONS diagrammatic method of representing construction systems is sufficient. These diagrams (an example of which is shown in Fig 2) form the basis of a model, representing the process flow and logic of the system under investigation. ICONS diagrams are developed from a suitable combination of the standard process symbols shown in Table I. The first two symbols are variant forms of the actqueue which is defined as a single operation, or discrete part of a process, undertaken by productive resources. The first type of actqueue represents an operation of a stationary productive resource such as a fixed batching plant, whilst the second is that of a moving productive resource such as a truck. Actqueueshave both an activity phase (representing the actual operation to be executed) which normally takes productive resources time to execute, and a queueing phase, in which productive resources wait while they are prevented from starting the activity.

The next two symbols in the figure are material nodes, representing a location in a system where material is transferred between productive resources. The hopper-type node has a facility for accumulating a buffer store of material between transferring productive resources. It can be used to represent, for example, a source of material input to a system or, an actual hopper/store such as a wet concrete hopper between a mixer and distribution trucks. The direct-transfer node, on the other hand, requires material to be transferred straight from one productive resource to another, making no provision for storage. An example is where an excavator loads spoil into trucks. The arrows on both types of material node indicate the direction of material transfer.

The final symbol is an arc which links successive actqueues illustrating the sequence of operations followed by productive resources. Usually, actqueues are linked into cycles, that is, the last in a sequence of actqueues is linked to the first of that group. The ICONS diagram shown in Fig 2 has two cycles of actqueues. The first represents the sequence of operations of an excavator and the second a fleet of trucks.

4. Forms of parallelism in ICONS models

There are a number of ways in which an ICONS model can be broken down into tasks for parallel execution on a multiprocessor. A fundamental problem is

to determine which of these forms of parallelism results in the most efficient implementation.

A simple approach is to allocate each cycle of actqueues in a model to its own processing element. During a simulation run, each processing element would check for, and execute, events associated with the operations of the productive resources within its allocated cycle. This would include:

a) Checking whether productive resources waiting in queues can start their activities;
b) generating activity durations;
c) checking to see whether productive resources have finished performing their current activities;
d) deciding to which actqueues productive resources should subsequently move (if there are several alternatives); and
e) ordering queueing productive resources.

Communication between processing elements is required when material is transferred between productive resources on different cycles. In the case of the excavation model illustrated in Fig 2, the processing element responsible for cycle C1, and that responsible for cycle C2, communicate when material is transferred at direct-transfer node N2,that is, when the excavator loads spoil into a truck. Although the time spent by processing elements whilst they are communicating is likely to be negligible, the overall efficiency of this type of implementation would be low. For many models, the number of tasks in parallel execution would be small and there would be a large imbalance in the workloads of processing elements. A processing element allocated a cycle performed by two productive resources would have approximately twice as much computation to perform as a processing element allocated a cycle performed by one productive resource. For the excavation model, there would only be two processing elements operating and that responsible for the trucks' cycle would have about five times the workload of that responsible for the excavator's cycle.

Another approach, likely to yield greater efficiency, would be to allocate each actqueue to its own processing element. A major drawback with this type of implementation, however, is that the level of efficiency would fluctuate markedly during a simulation run. The efficiency at each stage in a simulation would depend on the number of productive resources currently at each actqueue. For example, the highest degree of efficiency in the excavation model would occur when each of the five trucks is at a different actqueue - in this case, there would be six processing elements operating each with approximately the same workload. The worst case would occur when all the trucks are at the same actqueue, when only two processing elements would be operating, one of which would have about five times the workload of the other.

An alternative approach to implementing ICONS on a multiprocessor, would be to allocate each productive resource to its own processing element. This way, the number of tasks in parallel execution would be maximized (for the excavation model, there would always be six processing elements operating) and each processing element would have approximately the same workload. Processing overheadswould result, however, from the need for communication between processing elements when productive resources interact with each other. Overall,

this type of implementation would yield the highest degree of efficiency and is, therefore, the approach proposed for the multiprocessor-based simulator.

5. A prototype simulator

A multi-microprocessor developed in the Department of Electrical Engineering and Electronics at UMIST (the CYBA-M [11]) was used as the base of a prototype of the simulator. The CYBA-M is a shared memory multiprocessor (as illustrated in Fig 1) built from 15 INTEL 8080-A microprocessors.

Each microprocessor was programmed to act as a productive resource. For this purpose, a standard set of program routines was developed detailing the various processing tasks associated with a productive resource in the execution of a simulation. To increase efficiency, each microprocessor was given its own copy of these routines in its private memory.

The shared memory was used to store a description of the structure, logic and state of the particular ICONS model being run. This consisted of a complete set of the model attributes which define the specific characteristics of each component (such as the material capacity of a hopper-type node) and keep account of the state of each component as a simulation progresses (such as the number and order of productive resources in a queue). By storing the description of a model in the shared memory it was guaranteed that changes made to its state by one microprocessor would be communicated to all the other microprocessors.

A number of parallel operating digital clocks were added to the CYBA-M to execute the timing element in an ICONS simulation. One group of clocks were dedicated to measuring the time of the day, day of the week and week in simulation, whilst another group had the function of ensuring that productive resources finished performing activities at the correct times in a simulation. Each productive resource was allocated one of the latter types of clock. Whenever a productive resource started an activity, its microprocessor would calculate the activity duration and load this value into the clock. The clock would count from this value down to zero and then signal that the productive resource had finished the activity. The productive resource's microprocessor would then update the state of the model accordingly.

All clocks were designed to count in synchronization with a master clock to ensure that they would advance time at the same rate. This method of synchronization was chosen since it allows an interactive user to reduce the speed at which a simulation progresses to any slower scale of time (that of real-life if necessary) simply by changing the speed at which the master clock counts. By running a simulation at the slower speeds, a user can observe in detail the various processes occurring in the model.

6. Performance of the prototype simulator

The prototype simulator was evaluated by comparing its performance with two conventional serial-type implementations of ICONS made on a PRIME 750 machine (a powerful minicomputer) and, one of the microprocessors of the CYBA-M.

An initial set of experiments were made for a variety of modelling situations

starting with one actqueue performed by one productive resource, ranging through to fourteen actqueues each being performed by one productive resource. The results of these experiments are shown in Fig 3, where the horizontal axis indicates the number of actqueues in a model and the vertical axis indicates the period of time it took to advance a model by one state. It can be seen from this figure that the performance of the simulator, unlike that of the serial implementations, was independent of the size of a model.

The above set of experiments were repeated for two through to fourteen productive resources performing each actqueue. In the case of the simulator, where the experiments required the use of more than fourteen microprocessors, performance results were arrived at by extrapolation. The shaded regions of Fig 4 show under which of these modelling situations the performance of the simulator surpassed that of the serial implementations. It can be seen from these results that for larger models (where processing speed is most critical) the best performance was achieved by the simulator. Minor inefficiencies were incurred by the simulator, however, which increased with the number of productive resources performing each actqueue.

A final series of experiments undertaken on a variety of different sized models, showed the simulator's hardware timing structure to vary from 31 to 1284 times faster than the serial software timing routine of the INTEL 8080-A microprocessor implementation. This vastly superior performance was attributable to the use of dedicated hardware clocks which precluded the need for software to perform timing calculations.

The main implication of the performance test results is that the processing speeds of a parallel computing device does not limit the size of model that it is practical to run. An added advantage of the simulator, however, was that it enhanced a user's ability to control and monitor a simulation. A user could, for example, stop and start productive resources during a simulation run, simply by stopping and starting their respective microprocessors. This, coupled with the facility for varying the scale of time at which the simulation clocks counted, was found to be particularly useful for checking for errors in the design of a model and identifying likely bottlenecks in the system under investigation.

7. Conclusions and recommendations

Parallel processing is a viable means of procuring fast and efficient processng of construction simulation models. High performance as such is essential if simulation techniques are to be exploited to their full potential within the construction industry, particularly where studies require the use of large and highly detailed models and results are often required quickly.

The unit with the highest performance in the simulator was the hardware timing structure, being several orders of magnitude faster than a serial software timing routine. Ideally, the same level of performance should be realized in all units. It is recommended, therefore, that future work should be concerned with developing futher hardware units for the simulator, thus removing, or at least minimising, the need for software. A future simulator should, however, have the necessary hardware to run models with up to 100 productive resources, to cover most situations.

A current area of research in the Department of Building Engineering at UMIST, is concerned with the development of a graphics interface for ICONS, which has the objective of increasing the efficiency of both building and experimenting with models. This facility should be an integral part of the simulator. Work is required, therefore, to establish the best approach to implementing the graphics interface on a parallel computing device.

Acknowledgements

The authors acknowledge with gratitude the assistance and facilities provided by the Department of Electrical Engineering and Electronics and the Department of Computation at UMIST.

References

1 R Pilcher and I Flood, "The Use of Simulation Models in Construction" Proceedings of the Institution of Civil Engineers, Part 1 Design and Construction, 76, pp 635-652, (August 1984).

2 "ICONS: Interactive Construction Simulation, Users' Manual, Version 2.0", Department of Building Engineering, University of Manchester Institute of Science and Technology, (Unpublished).

3 D W Halpin and R W Woodhead, "Design of Construction and Process Operations", (John Wiley and Sons, New York, 1976).

4 R W Hockey and C R Jesshope, "Parallel Computers: Architecture, Programming, and Algorithms", (Hilger, 1981).

5 M J Flynn, "Very High-Speed Computing Systems", Proceedings of IEEE, 54, (12), pp 1901-1909, (December 1966).

6 A K Korn, "Back to Parallel Computation: Proposal for a completely new On-Line Simulation System using Standard Minicomputers for Low-Cost Processing", Simulation, 19, (2), pp 37-45, (August 1972).

7 J R Pimental, "Real Time Simulation using Multiple Microcomputers", Simulation, 40, (3), pp 93-104, (March 1983).

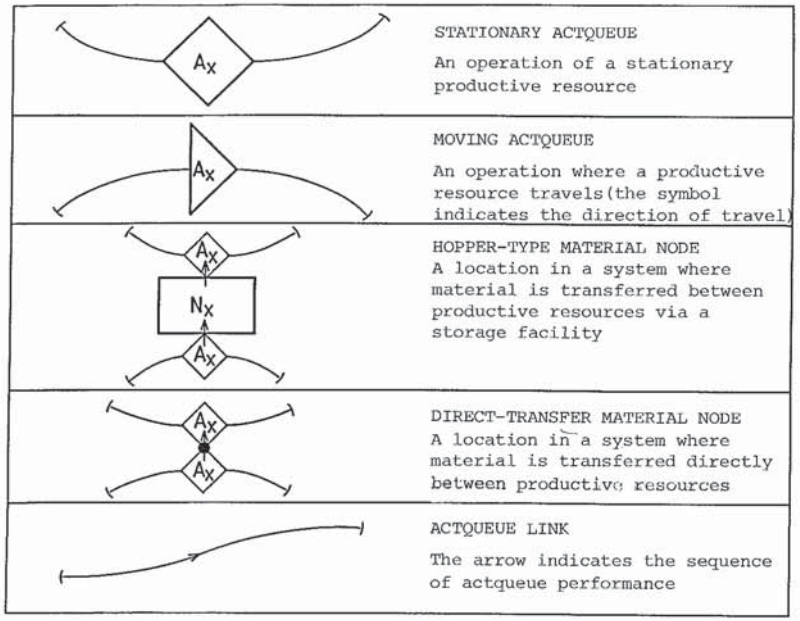
8 T Nakagawa, I Kumata, T Hasegawa, T Matsumoto, K Abe, N Kobayashi and H Aiso, "A Multi-Microprocessor Approach to Discrete System Simulation", Proceedings of the 20th IEEE COMPCON, (1980), pp 350-355.

9 L Dekker, "System Simulation and Parallel Data-Processing", Proceedings of the 1978 UKSC Conference on Computer Simulation, (Science and Technology Press, 1978), pp 84-106.

10 L Dekker, E J H Kerckhoffs, G C Vansteenkiste, and J C Zuidervaart, "Outline of a Future Parallel Simulator", Simulation of Systems '79, Proceedings of the 9th IMACS Congress, edited by L Dekker, G Savastano, and G C Vansteenkiste, (North-Holland Publishing Company, Amsterdam, 1980), pp 837-864.

11 D Aspinall and E R Dagless, "Overview of a Development Environment", Microprocessors and Microsystems, 3, (7), pp 301-305, (September 1979).

Table I Process symbols used for building ICONS diagrams



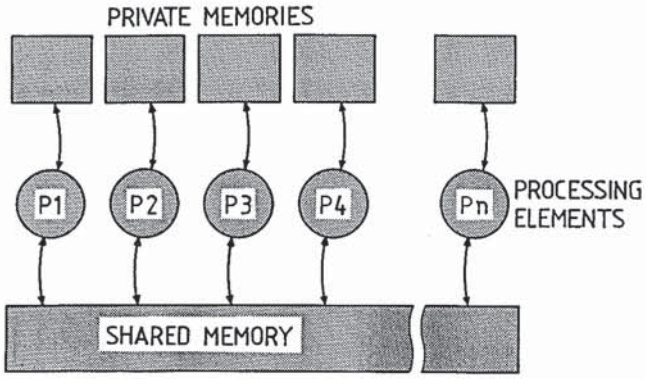


Figure 1 Outline of a shared memory multiprocessor

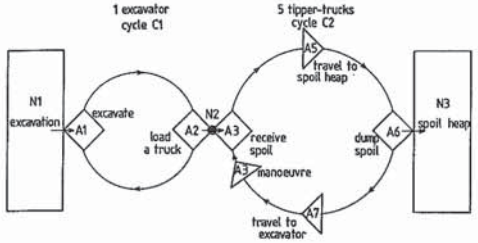


Figure 2 ICONS diagram of an excavation system

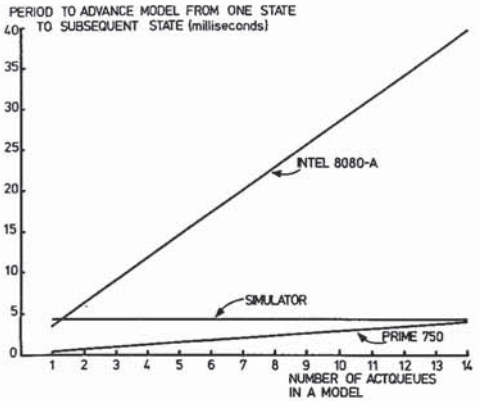


Figure 3 Performance results for models with one to fourteen actqueues, each actqueue being performed by one productive resource.

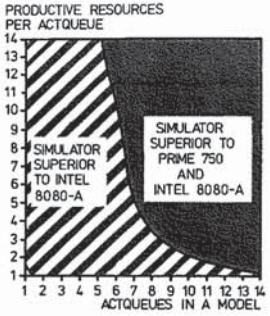


Figure 4 Modelling situation where the simulator's performance is superior to the two serial impelementations.

Computers and Construction Control

Axel Gaarslev

Technical University of Denmark
Department of Construction Management
Building 115
2800 Lyngby, Denmark

KEYWORDS

Construction Management, Construction Control, Computers, Site Management, Site Control.

ABSTRACT

Profound project control is often crucial to project success. The paper discusses how computers successfully are being used by contractors in this field and what the future probably will bring. Two different types of data are identified: Local data sets for a specific site in question and aggregated data sets for the division or company in question. Information is needed on both types of data on the different sites as well as at the main office, but the focus on site is probably on local data and in main office on aggregated data. This information structure is very complicated and will probably in the future technologically be solved by using intelligent terminals on sites tied to a main computer at the head office. Implementing such a system and making it work is a very complicated effort which most contractors probably can't accomplice in one step. Based on Danish experience two different implementation strategies are discussed, one based on a Top Down Strategy starting at main office and one based on a Bottom Up Strategy starting on site. Each strategy reflects it's own intermediate focus point, a top down strategy a central need for aggregated data, a bottom up strategy a local need for local data. Danish experience on these strategies are reported and recommendations on selecting implementation strategy are offered.