Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

A MULTI-BUS INTERCONNECTION MODEL

Baruch Halachmi

Advanced Simulation Concepts
P.O.Box 2308, Rockville, MD 20852

ABSTRACT

This paper presents a comparative analysis for the PxMxB multi-bus computer system. The system consists of a set of P processors that access a set of M shared memories via a set of B buses. We begin by describing the system and the underlying assumptions to model it. Then, an analytical algorithm is presented to approximate the throughput of the system. Next, simulation results are shown for a model that mimics the same set of assumptions used for the analytical approximation. Finally, we substantiate the merit of the simulation approach by illustrating the dramatic effect of removing "a minor simplifying assumption" from the simulation model, thus making the simulation model more realistic.

1. INTRODUCTION

Analyses of multi-processor computer systems attracted tremendous attention over the past two decades. Many computer architectures have been studied and their potential performance characteristics evaluated [Marsan 1986]. A typical performance analysis of a multi-processing system requires either a queuing model or a simulation model. When system characteristics are simple, an analytical queuing model may give us just what we need. An analytical solution provides not only the numerical means but also an insight into how parameters are interrelated in determining the estimated level of performance. When a system is too complex for analytical handling, our next bet is the simulation approach [MacDougall 1987] and [Law 1991]. Via a simulation approach, unnecessary simplifying assumptions may be eliminated. Moreover a simulation model may capture the dynamics of a system with a great level of detail.

In this paper we analyze the performance of the basic PxMxB multi-processing system. We present an analytical model and then compare it numerically against two sets of simulation results.

2. OVERVIEW

Consider a multi-processor architecture with:

P - identical parallel processors,

M - equally referenced memories, and

B - global buses

Every processor may access at a given time any of the M memories using one of the B buses as depicted in Figure 1

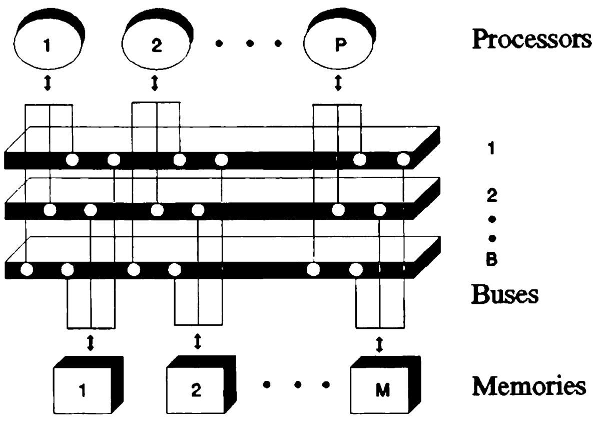

Figure 1. A multi-processing system

In a given machine cycle a processor generates a memory request (i.e. a **store** or a **fetch**) with a probability of Θ. If the addressed memory is free and a bus is available, the request is satisfied in the next cycle. If however another processor places a request to that memory in the same cycle or a bus is not available, the request is reintroduced in the following cycle until it gets satisfied. During the wait for the memory request to be completed the issuing processor is **blocked**.

3. ASSUMPTIONS

(a) In a given cycle only one out of multiple requests to a given memory may be serviced. Service time is one cycle.

(b) At a beginning of a cycle a request may be issued to a memory only if one of the buses is available. Therefore, if B < M, at the most B requests are serviced simultaneously.

(c) A request satisfied in a given cycle enables the processor to continue its execution at the following cycle. If however the request is **blocked** because the memory is engaged in servicing another request, or all buses are busy, the request is reissued in the next cycle.

(d) After every cycle all buses are disengaged from any processor/memory association.

4. ANALYTICAL APPROXIMATION

The analysis presented here is very similar to the one outlined in [MacDougall 1987] and [Mudge 1984]. The main difference is the introduction of the $q_{r,k}$ probabilities at 4.2.

(a) Assume initially that at the beginning of a cycle there are no outstanding requests from previous cycle(s).

4.1 The probability $\mathbf{P_r}$ of having **r** requests in a given cycle is given by:

$$P_r = \binom{P}{r}\Theta^r(1-\Theta)^{P-r}, \qquad r=0,1,\ \dots\ P$$

4.2 The probability $\mathbf{q_{r,k}}$ that in a given cycle **r** requests are addressed to exactly **k** memories is given by:

$$q_{r,k} = \frac{\binom{M}{k}\binom{r-1}{k-1}}{\binom{r+M-1}{M-1}}, \qquad k=1,\ \dots M, \qquad r=k,\dots P$$

4.3 The probability π_k that in a given cycle exactly k memories are referenced is therefore:

$$\pi_k = \sum\nolimits_{r=k}^{P} P_r q_{r,k}\ , \qquad k=1,\ \dots\ M$$

4.4 The BandWidth of the system **BW(p)** is the rate that the system completes memory requests. It equals:

$$BW(\Theta) = \sum\nolimits_{k=1}^{M} \min(k,B)\cdot\pi_k$$

(b) Assume now that at the beginning of a given cycle there are also requests that were not satisfied in previous cycles.

4.5 We apply here the approximation **BW(α)** for the BandWidth, where $\Theta \le \alpha$. It is thus implied in the approximation for the BandWidth that colliding requests **only** increase the effective rate of emitting memory requests. In reality though a colliding request is addressed in the next cycle to the same memory. Thus if for example in a given cycle 3 requests are issued to a given memory, in the following cycle at least 2 requests are posted for that memory. The analytical approximation does not take this dependency into account. It assumes uniform distribution of requests at the beginning of every cycle.

4.6 Denote by **c** the average number of cycles before a request is issued. The geometric distribution yields:

$$c = 1/\Theta - 1$$

4.7 The average time **T**, between the completion of consecutive requests at the individual processor, is given by:

$T = P / BW(\alpha)$

4.8 With α, c and T we have the relationship:

$1-\alpha = c / T$.

4.9 From 4.6, 4.7 and 4.8 it follows:

$\alpha = 1 - c \cdot BW(\alpha)/P$

4.10 Although 4.9 provides a relationship between α and $BW(\alpha)$, it is hard to evaluate from it $BW(\alpha)$ directly. Alternatively, we apply here the **Fixed Point Iteration** method [Conte 1972]. In order to guarantee a convergence, 4.9 is rewritten as:

$$\alpha_i = \left[1+\frac{c \cdot BW(\alpha_{i-1})}{P \cdot \alpha_{i-1}}\right]^{-1}, \quad i=1,2,\ \dots \qquad \alpha_0=\theta$$

4.11 Once α^* is obtained via 4.10, the ThroughPut **TP** of the system is approximated as:

$TP = P \cdot (1 - \alpha^*)$

5. THE SIMULATION MODEL

5.1 GENERAL

We built the simulation model with Emula4 [Halachmi 1993]. Emula4 is a process oriented network simulation language designed for the MS-WINDOWS platform. It is a multi-tasked simulation superset of the standard programming language "C" [Kernighan 1988].

5.2 THE SIMULATION MODEL

The main logic of the simulation model consists of two types of processes, **Fetch** and **Execute**. At the simulation start a pair of these processes is spawned for each of the P CPUs.

5.2.1 THE FETCH PROCESS

The Fetch process contains the logic of processing machine instructions at a given CPU. Every once in a while a memory request is generated and the CPU is blocked until a response is received for this memory request.

5.2.2 THE EXECUTE PROCESS

An Execute process serves all memory requests for a given CPU. When a memory request registers at an Execute process, the addressed memory is checked. If busy, the request is delayed for a cycle. Likewise, if a bus is not available, the request is not processed in that cycle. This wait may last several cycles until both a memory and a bus are found available to carry out the request. At this point the memory and the bus are seized for a cycle, and then a response is sent back to the corresponding Fetch process, unblocking the CPU to continue processing machine instructions until the next memory request is issued.

5.2.3 THE FETCH & EXECUTE CODE

```
process Fetch(int cpu)
{
   int  request, response;

   while(TRUE)
   {
      delay(geometric(Theta));
      request = memory(M);
      putpkt(cpu2mem[cpu], request);
      getpkt(mem2cpu[cpu], response);
   }
}

process Execute(int cpu)
{
   int  request, response;

   while(TRUE)
   {
      getpkt(cpu2mem[cpu], request);
      while( (MEM[request] == BUSY) ||
            (AvailBus == 0) )
      delay(CYCLE);
      MEM[request] = BUSY;
      AvailBus--;
      delay(CYCLE);
      MEM[request] = !BUSY;
      AvailBus++;
      putpkt(mem2cpu[cpu], response);
   }
}
```

The model was first coded with the same set of assumptions as described for its analytical counterpart. Next we modified it (as above) to accommodate the more realistic assumption that if a request is not satisfied in a given cycle it is always reissued to the same memory unit.

6. NUMERICAL RESULTS

The simulation model was executed with the number of processors varying between 1 and 10. For every run we measured the system throughput and compared it with the analytical approximation derived via 4.10 and 4.11.

Two sets of results were generated:

I. corresponds to the system with the same assumptions as applied for the analytical model.

II. corresponds to the system without the simplifying assumption about memory references.

The numerical results are presented in Table 1 and Figure 2. It is apparent from these results that for a small number of processors (relative to the number of memories) the analytical formula approximates well the throughput of the system. As the number of processors grows, the increase in throughput diminishes.

Also notice that the analytical approximation overshoots the anticipated values, especially of set II. **The analytical formula deteriorates partially due to the essential (but unrealistic) assumption that if a request to a given memory is not satisfied in a cycle, the request is reissued in the subsequent cycle, but not necessarily to the same memory unit.** There is no need of course to commit to such a simplifying assumption in the simulation model.

INPUT: $B = 2;$ $P = 1, \dots 10;$
$M = 4;$ $\Theta = 0.25;$

CPUs #	Anal. Appr.	Sim. set I	Sim. set II	Max. Val.
1	0.75	0.75	0.76	0.75
2	1.48	1.48	1.49	1.50
3	2.18	2.19	2.20	2.25
4	2.85	2.87	2.87	3.00
5	3.48	3.45	3.41	3.75
6	4.05	3.93	3.90	4.50
7	4.55	4.35	4.32	5.25
8	4.98	4.85	4.70	6.00
9	5.31	5.11	4.96	6.75
10	5.54	5.18	5.06	7.50

Table 1. Estimated Throughput values

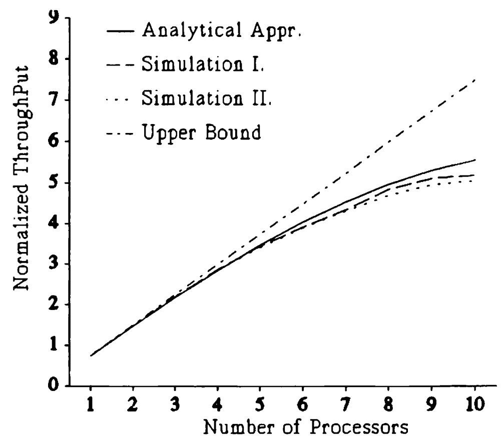

Figure 2. Estimated throughput curves

7. CONCLUSION

Analytical models and simulation models work side by side. They are complimentary to each other. In the case of the simple interconnection bus system, the iterative analytical procedure provides quick and valuable estimates. When we need to analyze a more detailed system, and cannot afford to make gross simplifying assumptions, the simulation approach is an ultimate choice.

REFERENCES

Performance Models of Multiprocessor Systems,
M. Marsan, G. Balbo, G. Conte, The MIT Press, 1986.

Simulating Computer Systems
M. MacDougall,
The MIT Press, 1987.

Simulation Modeling and Analysis,
A. Law, W. Kelton,
MacGraw-Hill, 1991.

Anlysis of Multiple Bus Interconnection Networks,
T. Mudge, J. Hayes, G. Buzzard, D. Winsor,
Proc. Intn'l Conference on Parallel Processing, 1984.

Elementary Numerical Analysis,
S. Conte, C. de Boor,
MacGrew-Hill, 1972.

Emula4 - A Network Simulation Language,
Baruch Halachmi,
Advanced Simulation Concepts, 1993.

The C Programming Language,
B. Kernighan, D. Ritchie,
Prentice Hall, 1988.

AUTHOR BIOGRAPHY

BARUCH HALACHMI is a communications/simulation consultant for Advanced Simulation Concepts. He received a B.A. degree in statistics from the Hebrew University of Jerusalem in 1968, M.A. degree in statistics and operations research from Tel-Aviv University in 1972, and a Ph.D. degree in computer science from the University of Minnesota in 1974. His areas of expertise include design and modeling of communication networks, simulation of computer imbedded systems, and software development of real time systems.