A PERFORMANCE COMPARISON OF OPTIMALLY DESIGNED COMPUTER SYSTEMS WITH AND WITHOUT VIRTUAL MEMORY

Kishor S. Trivedi* and Timothy M. Sigmon**

Department of Computer Science
Duke University
Durham, North Carolina 27706

ABSTRACT

In this paper, a comparison of the performance of optimally designed computer systems with and without virtual memory is made. The computer systems in question are modeled by closed queuing networks of the central server type. The design of the systems is formulated as a nonlinear optimization problem where the objective function is to maximize the throughput subject to a nonlinear cost constraint. The decision variables are the speeds of the individual devices. This optimization problem is then solved by use of the Lagrange multiplier technique. The comparisons of the systems demonstrate the affect on performance of the addition of another I/O device to handle paging and the affect on performance of the additional overhead generated by the page fault handler. Also, the optimal amount of money to be spent on main memory is investigated.

I. Introduction

Recently, an increasing amount of effort has been directed toward the development of models to aid the computer system designer in his work. Of particular concern in this paper are the efforts in the area of the optimization of closed queuing networks which are suitable for modeling multiprogrammed computer systems [1,2,3]. Trivedi and Wagner [4] have recently shown that the problem of maximizing the throughput in a closed queuing network subject to a rich class of nonlinear cost constraints has only one local maximum which is, therefore, the global optimum. They also show that the cost constraint must be active at the solution which enables the problem to be efficiently solved by use of a Lagrange multiplier technique [5]. Trivedi and Kinicki [6] exploit the above results and provide several examples of how a system designer might select device speeds by using this technique. Although the speeds of the devices will be discrete in real life, our model will approximate them to be continuous variables. Due to this and other assumptions, solutions obtained by our model should be considered initial approximations in an essentially iterative design process.

Section 2 of this paper will briefly describe the model used by Trivedi and Kinicki and will then describe the modifications necessary to allow the modeling of virtual memory. Section 3 will describe the parameterization of the models and will then discuss the results of the comparisons of the computer systems with and without virtual memory.

II. Description of the Models

We first describe the model of a computer system operating in a multiprogramming environment without virtual memory where each active program's address space resides in main memory until its completion. The system is represented by a closed queuing network of the central server type [7] (see figure 1). Each significant device in the system is represented by one of the m+1 service facilities (numbered 0 to m) where the i-th service facility consists of a queue and a single server which is capable of processing b(i) work units per unit time. All service facilities are assumed to operate under a FCFS scheduling discipline and their service time distributions are assumed to be negative exponential with mean service rate u(i) (i = 0,1,...,m). Any differentiable service time distribution is permitted if the scheduling discipline is PS or LCFS-PR [8]. The network contains n (the degree of multiprogramming) stochastically equivalent programs which alternate between service at the CPU (device 0) and one of the m I/O devices according to branching probabilities which characterize the movement of a typical program through the computer system. p(0) is the probability that upon completion of service at the CPU the program terminates and a new program enters the system via the new program path. By assuming that there is a sufficiently large backlog of programs awaiting service, a departure from the system triggers the instantaneous arrival of a new program which maintains the degree of multiprogramming at n. The throughput of the

* Supported in part under the National Science Foundation grant number US NSF MCS 76-24417.

** Supported in part under the National Library of Medicine Training grant number LM-07003.

CH1394-6/79/0000-0117 $00.75 ©1979 IEEE

system can then be easily described as the rate of flow along the new program path. p(i) for i = 1,2,...,m is the probability that upon leaving the CPU a program will next require service at the i-th I/O device.

Let t(i) be the average number of program visits to the i-th facility, and let J(i) be the total number of work units processed per program at the i-th facility. Thus, J(0)/t(0) represents the average number of instructions executed by the CPU between two I/O requests and J(i)/t(i) (i = 1,2,...,m) is interpreted as the number of information units transferred between the i-th device and main memory per visit. Now the speed of the i-th service facility can be expressed as a function of the service rate as follows:

$$b(i) = u(i)*\frac{J(i)}{t(i)} \qquad i = 0,1,\ldots,m \; . \qquad (1)$$

The cost of the computer system is approximated by the sum of the individual component costs and the main memory cost. The cost of each component is expressed in terms of a continuous power function of the device speed and the cost of main memory is assumed to be a linear function of n.

The Trivedi-Kinicki design problem can now be expressed as the following optimization problem:

$$\text{maximize} \qquad T(b(0),b(1),\ldots,b(m)) \qquad (2)$$

$$\text{subject to} \qquad \sum_{i=0}^{m} C(i)*b(i)^{alpha(i)} + Mem(n) \leq COST$$

$$\text{and} \qquad b(i) > 0 \qquad i=0,1,\ldots,m$$

where T(b(0),b(1),...,b(m)) is the throughput of the system. In words, the design problem is: given a fixed topology, a fixed degree of multiprogramming, a workload description (in terms of fixed branching probabilities and J(i), t(i) for i = 0,1,...,m), and the amount of main memory required per program (equal to the size of its entire address space), determine the device speeds which maximize the system throughput subject to a cost constraint. Trivedi and Kinicki then use a Lagrange multiplier technique to solve the above problem for several examples. (Since the degree of multiprogramming is an integral variable, a discrete search is performed to find the optimal degree of multiprogramming.)

Based upon the above model of a computer system without virtual memory, we now develop the two basic models of a computer system with virtual memory which will be used in the comparisons of section 3. The difference between these two models is the method of implementing the virtual memory which yields two different system topologies. In one model, another I/O device is added to the system to handle the paging traffic, while in the other model, all paging I/O is handled by one of the existing I/O devices whose capacity has been increased. As far as the mathematical and computational aspects are concerned, the only difference between either one of the virtual models and its topologically equivalent non-virtual model is in the characterization of the workload. In particular, the branching probabilities in the virtual models are functions of the degree of multiprogramming, n, instead of being fixed. This can be easily shown as follows.

The total I/O activity of the system consists of two parts - paging I/O and all other I/O. The average CPU burst between two paging I/O requests is given by the system lifetime function e(mem), where mem is the amount of main memory allocated to each program in the active set. Note that this lifetime function is clearly a function of n since mem = M/n where M is the total amount of main memory. The average CPU burst between two non-paging I/O requests is w which is given by $w = J(0)/(\sum t(i) + 1)$ where the index of the summation, i, is varied over all non-paging I/O devices. Note that w does not depend on n. Combining the above two expressions yields E(n), the average CPU burst:

$$\frac{1}{E(n)} = \frac{1}{e(n)} + \frac{1}{w}$$

or

$$E(n) = \left(\frac{1}{e(n)} + \frac{1}{w}\right)^{-1} \; . \qquad (3)$$

Since J(0) is the total number of instructions to be executed per program, then t(0) = J(0)/E(n). Realizing that p(0) = 1/t(0) yields p(0) = E(n)/J(0). Also the branching probabilities for all non-paging I/O devices are given by p(i) = t(i)/t(0) = p(0)*t(i) where i = indices of all non-paging I/O devices. The branching probability for the paging I/O device can then be easily found by summing all known probabilities and subtracting from one. Thus we see that the branching probabilities in the virtual models are functions of the degree of multiprogramming. This does not affect the mathematical optimization problem, however, since it is assumed that the degree of multiprogramming is fixed.

One additional feature of the virtual models is the inclusion of a term to account for the CPU overhead generated by the page fault handler. This is accomplished by modifying J(0) as follows:

$$J(0)' = J(0) + \left\lceil \frac{J(0)}{e(n)} \right\rceil *PFH \qquad (4)$$

where $\left\lceil \frac{J(0)}{e(n)} \right\rceil$ is the number of page faults that

were generated and PFH is the number of instructions executed by the page fault handler.

III. Results of the Comparisons

The comparisons of computer systems with and without virtual memory will be based on the following three models: 1) a multiprogrammed computer system without virtual memory and having three I/O devices, 2) a multiprogrammed computer system with virtual memory and having four I/O devices one of which is dedicated to paging I/O, and 3) a multiprogrammed computer system with virtual memory and having three I/O devices one of which handles both paging and non-paging I/O.

The model of the non-virtual computer system will be parameterized exactly as it was in [6]. These parameters are displayed in table 1. Tables 2 and 3 display the parameters of the two virtual models. Belady's lifetime function, $e(mem) = a*mem^b$, was chosen to represent the length of the CPU burst between page faults. The values for a (4.69) and b (2.88) were obtained from Spirn [9] who provided a fit of Belady's lifetime function based on empirical data. All of the pricing information is the same as that used by Trivedi and Kinicki except for the cost coefficient of the drum in table 3 which was doubled to reflect the increased capacity required by its double duties. As was done in [6], the price of main memory was approximated by $1.00 per 32-bit word. The total system budgets that will be considered range from $1,000,000 to $2,000,000 in increments of $250,000. One final parameter concerns the amount of main memory required by each program in the active set in the non-virtual model. This value will be assumed to be 50,000 words.

Figure 2 is a graph of optimal throughput versus dollars spent on main memory for the non-virtual model and the virtual model with four I/O devices. The dashed lines are the results from the non-virtual model and the solid lines are the results from the virtual model with four I/O devices. The results from two total system budgets and three values of page fault handler overhead are plotted on the same graph. Each point of the virtual model's curves was obtained by choosing the optimal point after a discrete search over n, the degree of multiprogramming, was performed. The small numbers written beside each point of the virtual model's curves are the optimal degrees of multiprogramming. The degree of multiprogramming for each point of the non-virtual model's curves is not given since it is easily obtained from n = $MM/50000 where $MM is the amount of money spent on main memory. It can be easily seen that when the page fault handler overhead (PFH) equals 10000 instructions the virtual curve lies completely below the non-virtual curve and when PFH = 0 the virtual curve lies completely above the non-virtual curve. When PFH = 5000, however, the virtual curve lies above the non-virtual curve for small amounts of main memory and below for larger amounts of main memory. This figure also clearly displays the increased degree of multiprogramming possible with virtual memory.

Figure 3 is a similar graph for the non-virtual model and the virtual model with only three I/O devices. Here it is easily seen that for all three values of PFH, the virtual curve lies completely above the non-virtual curve. Thus, for this set of values for the model parameters, we conclude that virtual memory will yield a performance increase when the paging I/O is handled by an existing I/O device.

Figure 4 is a graph of optimal throughput versus total system budget for all three models. The dashed curve again represents the non-virtual model while the two solid curves representing the virtual models are labeled. In this figure, the page fault handler overhead is 10000 instructions for both virtual models. Associated with each point on the graph is a pair of numbers which specify the optimal degree of multiprogramming and the optimal amount of money to be spent on main memory (in thousands of dollars). It is assumed that money is allocated to main memory in increments of $50,000. This type of graph clearly shows the affect that an increased system budget has on the optimal degree of multiprogramming and on the amount of money that should be spent on main memory.

IV. Conclusion

The examples in this paper demonstrate a mathematical tool which could prove useful in aiding the system designer in his decision of whether to use virtual memory and, if so, the appropriate method of implementing it. This paper does not advocate one method of implementation over another since this is dependent on the system workload and costs by which the designer is currently constrained. A particular advantage of this tool is that it provides a simple and inexpensive method of gaining insights about a large number of different system configurations operating under varying workloads and constrained by different cost estimates.

REFERENCES

[1] Chiu, W.W.-Y. Analysis and applications of probabilistic models of multiprogrammed computer systems. Ph.D. Dissertation, Department of Electrical Engineering, University of California, Santa Barbara, California, December 1973.

[2] Hogarth, J. Optimization and analysis of queueing networks. Ph.D. Dissertation, Department of Computer Science, University of Texas, Austin, Texas, May 1975.

[3] Kachhal, S.K. and Arora, S.R. Seeking configurational optimization in computer systems. ACM Annual Conference (1975), pp. 96-101.

[4] Trivedi, K.S. and Wagner, R.A. A decision model for closed queuing networks. Accepted subject to revision, IEEE Transactions on Software Engineering.

[5] Luenberger, D.G. Introduction to Linear and Nonlinear Programming. Addison-Wesley Publishing, Reading, Massachusetts, 1973.

[6] Trivedi, K.S. and Kinicki, R.E. A Mathematical Model for Computer System Configuration Planning. Proc. International Conference on the Performance of Computer Installations, D. Ferrari (ed.), North Holland, Amsterdam, 1978.

[7] Buzen, J.P. Computational algorithms for closed queueing networks with exponential servers. CACM 16, 9(September 1973), pp. 527-531.

[8] Chandy, K.M., Howard, J.H., and Towsley, D.F. Product form and local balance in queueing networks. JACM 24, 2(April 1977), pp. 250-263.

[9] Spirn, J.R. Program Behavior: Models and Measurement. Elsevier North-Holland, N.Y., N.Y., 1977.

i	Device Name	J(i)	t(i)	p(i)	C(i)	alpha(i)
0	CPU	400,000	20	0.05	1,147,835.00	0.55309
1	Drum	10,000	10	0.50	1,432,664.00	1.00000
2	Disk 1	6,000	6	0.30	707,648.00	0.67290
3	Disk 2	3,000	3	0.15	707,648.00	0.67290

Table 1 - without virtual memory (m = 3)

i	Device Name	J(i)	t(i)	C(i)	alpha(i)
0	CPU	400,000	*	1,147,835.00	0.55309
1	Paging Drum	*	*	1,432,664.00	1.00000
2	Drum	10,000	10	1,432,664.00	1.00000
3	Disk 1	6,000	6	707,648.00	0.67290
4	Disk 2	3,000	3	707,648.00	0.67290

* these values are dependent on n and the lifetime function

Table 2 - with virtual memory (m = 4)

i	Device Name	J(i)	t(i)	C(i)	alpha(i)
0	CPU	400,000	*	1,147,835.00	0.55309
1	Paging Drum	*	*	2,865,328.00	1.00000
2	Disk 1	6,000	6	707,648.00	0.67290
3	Disk 2	3,000	3	707,648.00	0.67290

* these values are dependent on n and the lifetime function

Table 3 - with virtual memory (m = 3)

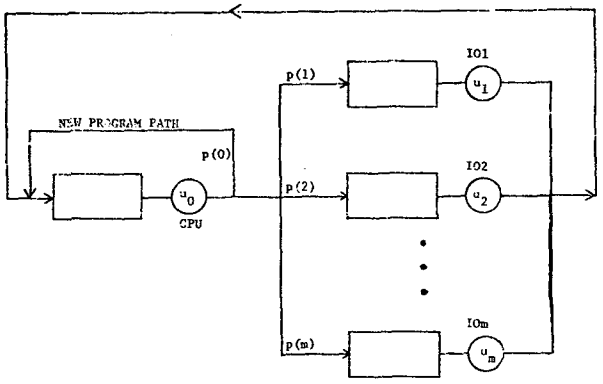

Figure 1 - Central Server Model

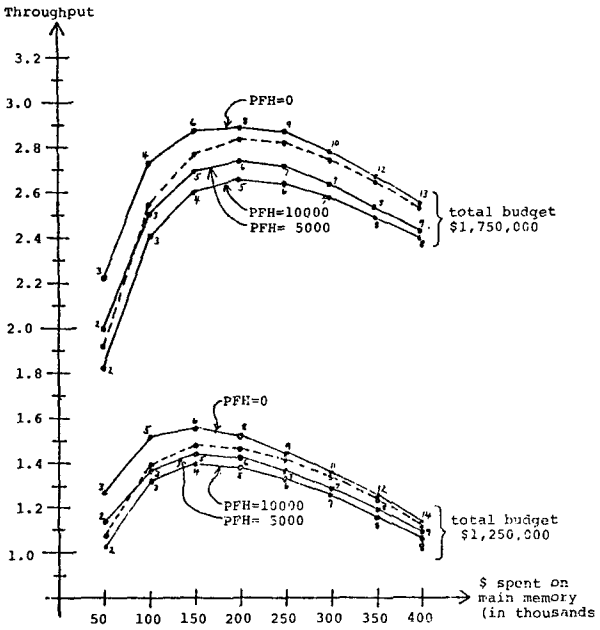

Figure 2 - Throughput vs. $ spent on main memory for the non-virtual model (dashed lines) and the virtual model with 4 I/O devices.

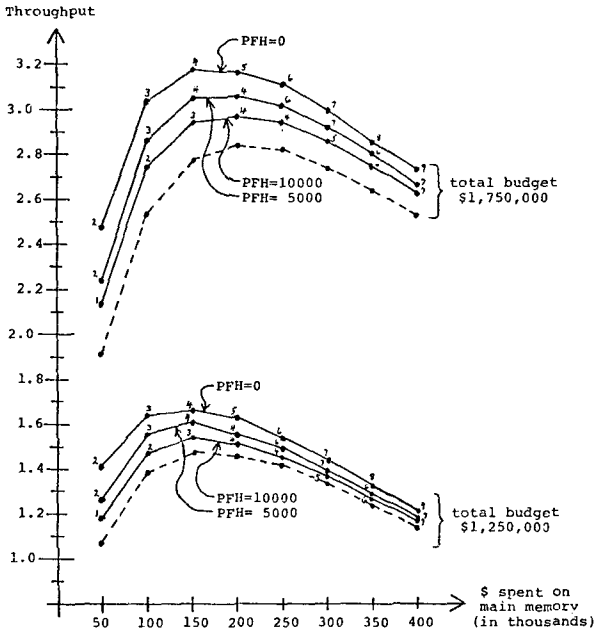

Figure 3 - Throughput vs. $ spent on main memory for the non-virtual model (dashed lines) and the virtual model with 3 I/O devices.

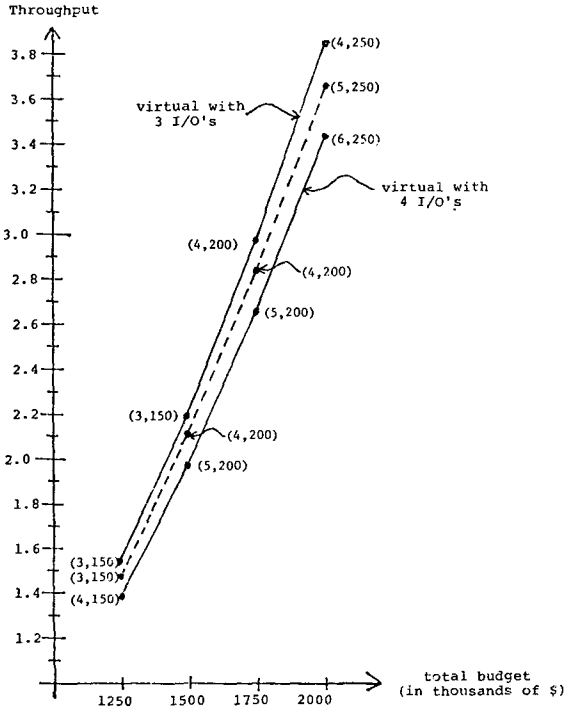

Figure 4 - Throughput vs. total budget