Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

DECISION MAKING OF EMBEDDED I/O BUFFER SIZES USING THE QUEUEING SIMULATION MODEL FOR A SHARED-MEMORY SYSTEM

Jui-Hua Li
JoAnne Holliday

Computer Engineering Department
Santa Clara University
Santa Clara, CA 95053, U.S.A.

George Fegan

Applied Mathematics Department
Santa Clara University
Santa Clara, CA 95053, U.S.A

ABSTRACT

This paper presents a methodology of decision-making for embedded I/O buffer sizes in a single-bus shared-memory system. The decision is made with the aid of a queuing model, simulation, and the proposed algorithm. The generalized queueing model is simulated to cover two cases: independent processing units and pipelined processing units in a shared-memory environment. The objective is to obtain the best performance with the optimized embedded buffers in the system. Therefore, an algorithm is developed to find the optimal solution efficiently by exploring the correlation between buffers and system performance. The local optimum is guaranteed. The method can be widely applied to many applications.

1 INTRODUCTION

In most cases of computational devices, memory is shared among different processing units (PUs) through a common bus. For intensive data processing systems such as multimedia system, MPEG encoder/decoder (Li and Ling 1999) (Lauzon, Vincent and Wang 1996) and local network, bus utilization and system performance are very important issues. The factors that affect system performance can be the number of processing units, memory type and bus structure, arbitration schemes, and workload (Kornecki and Zalewski 1998) (Jonkers 1994). In this paper, we focus on the factor of embedded I/O buffers of the processing units, which has not been explored in detail in prior works.

In a shared-memory system, the time to access the data in memory involves waiting time, switching time and data transfer time. Data transfer time and switching time for one transfer are determined by the memory and bus structure. The total switching time is proportional to the number of bus requests issued during the processing of one task. The processing unit with smaller I/O buffer requests more data transfers. The proportion of switching time to total access time can be significant if the I/O buffer is small. On the other hand, if the I/O buffer is too big, it causes delay for other processing units in accessing the bus. To maintain the system at the highest performance or best utilization of hardware resources, the buffer sizes need to be properly chosen.

In this paper, a generalized queueing model is presented for two scenarios: independent processing units and pipelined processing units in the system. A simulation program, based on this model, was developed to analyze traffic patterns and to evaluate the system performance. The algorithm searches the decision space for the solution yielding the best performance, i.e., the best buffer setting.

The approach is outlined as follows:

1. Creating a general queueing model for the real application.
2. Formulating the problem and constructing a mathematical generalized model to represent the system (Hillier and Lieberman 1995). Identifying the objectives, decision variables/random variables, and constraints in the problem.
3. Simulating the model and applying the proposed algorithm to search the decision space, i.e., all the possible solutions, to yield the optimal solution.

2 QUEUEING MODEL FOR A SINGLE-BUS SHARED-MEMORY SYSTEM

A generalized shared-memory system is illustrated in Figure 1. Processing units are independent if data in memory are not shared among them. In figure 1, the arrows indicate independent data flows. Processing units are dependent if processing units cooperate to process data in pipeline manner, as shown by the dashed data flow in figure 1. Each Processing unit (PU) has its I/O buffer to temporarily store data from/to memory. A bus request is issued whenever the buffer is either empty, to receive new data, or when full, to transmit processed data. The data is transferred from memory, through a common bus, to the buffers in a fixed size.

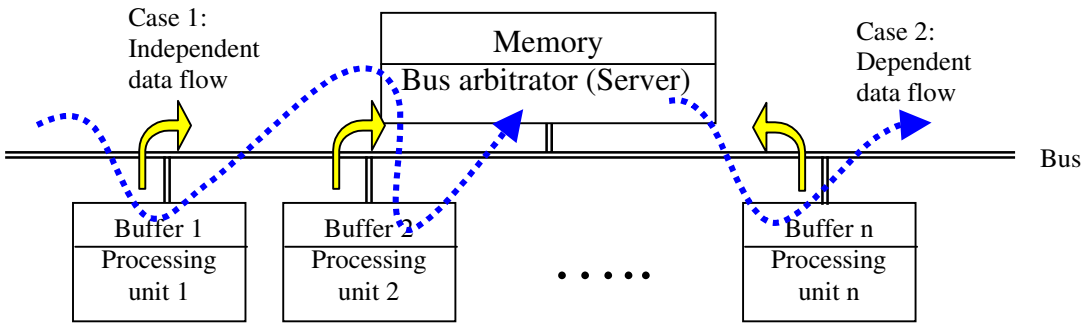

Figure 1: A Generalized System Model

The amount of data in a transfer is usually the same as the size of the buffer. The traffic on the bus is determined by the application and the buffer size is determined by a system designer.

In Figure 2, the whole system is presented from the bus administrator's point of view. Each arrival represents a request for the bus. Arrivals can be classified into random and deterministic types due to the variance of data consuming rates on the buffers. Arrivals are correlated in the case of pipelined-PU model, otherwise they are independent. Data transfer is done in the service station. The waiting capacity for the service is limited. Service time is deterministic. Queueing disciplines include FCFS (First Come First Served) and priority schemes.

The service time is deterministic for each PU because it only depends on the buffer size and bandwidth of the bus, which are both determined by the designer. We consider a queueing system where some of the bus requests have a deterministic arrival time and some have a random arrival time. Furthermore, the buffer size is different for each PU, and therefore the pattern of the service time is no longer deterministic.

3 FORMULATION

In this section, the system is reviewed to identify the objective and variables. The objective is to achieve the best system performance with the optimal buffer setting. The I/O buffer sizes are variables which need to be determined. Thus the system is formulated in a mathematical way for the optimization. The pattern of bus requests issued from a PU may be described by a statistical function or a deterministic type of function. Given the statistics of the bus requests, the exact relation between buffer size, switching time, and waiting time must be determined in order to find the optimal buffer size and maximum system throughput. For instance, the waiting time for each PU is affected by the data requests over the bus from other PUs. The occurrence of each arrival depends on the data-consuming speed for the buffer. The consuming speed is determined by the hardware and buffer size. Hence, we can describe the expected waiting time as a function of the buffer lengths for the PUs, statistics of data request patterns and data processing rates.

$$
\begin{aligned}
W_1 &= f_1(X_1, X_2, \ldots, X_n, V_1, V_2, \ldots, V_n, R_1, R_2, \ldots, R_n) \\
W_2 &= f_2(X_1, X_2, \ldots, X_n, V_1, V_2, \ldots, V_n, R_1, R_2, \ldots, R_n) \\
W_3 &= f_3(X_1, X_2, \ldots, X_n, V_1, V_2, \ldots, V_n, R_1, R_2, \ldots, R_n) \\
&\vdots \\
W_n &= f_n(X_1, X_2, \ldots, X_n, V_1, V_2, \ldots, V_n, R_1, R_2, \ldots, R_n) \\
&\text{for } i=1, \ldots, n
\end{aligned} \tag{1}
$$

where

- W_i is the expected waiting function for PU_i.
- X_i is the buffer size.
- V_i is the characteristics of data described by statistics distributions.
- R_i is the data processing speed for the processing unit.
- n is the number of PUs.

Since The total switching time is proportional to the number of bus requests issued during the processing of one task, the switching time can be derived easily from definition. We sum up switching time, data transfer time and waiting time to gain total memory time. The best performance is achieved at minimal total time. In other words, we

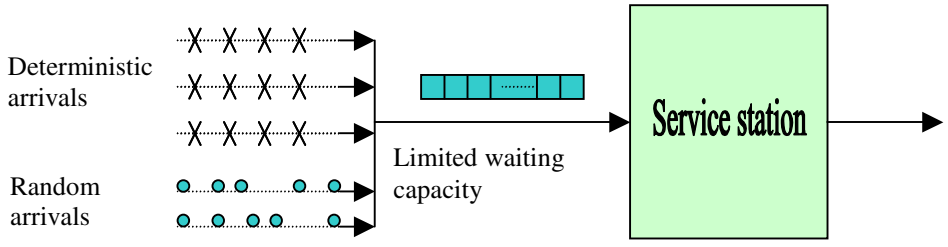

Figure 2: Bus Administrator

would like to define the objective function as system performance *P*. *P* can be the throughput of the system. Assume the hardware features like processing rates and statistics of bus requests are determined, then *P* becomes the function of "buffer" variables as follows.

$$P = f(X_1, X_2, X_3, \ldots X_n)$$
$$\text{where } X_i \text{ is the buffer size for } PU_i;\ i = 1, \ldots, n \qquad (2)$$

Beyond the formulation, a simulation is needed due to the difficulty of deriving a closed form solution for a complex system. It is not hard to imagine the complexity of the generalized model, which involves more random-type arrivals and constraints. Determining the appropriate objective function is also a critical and challenging part of the model-building process. Therefore, simulation becomes important and relevant.

4 PROPOSED ALGORITHM: THE OPTIMAL I/O BUFFER SETTING

Formulation: Given the system above with fixed hardware configuration. Let *P* be the throughput of the system; $P = f(x_1, x_2, \ldots, x_n)$, and *B* be the bus utilization; where x_i is the size of the i'th I/O buffer in the system; $x_i \in \mathbf{S}_i$, i=1, ..., n, and $\mathbf{S}_i$ is the set of all possible buffer sizes. $\varepsilon>0$ is the stopping criteria.

Optimize $P = f(x_1, x_2, \ldots, x_n)$ subject to $P > c$ and $B < k$ for some constants positive c and k. *c* is the minimum throughput required by the system. *k* is the maximum bus bandwidth of the system.

Algorithm: **Initialization step**: Set all buffers to their smallest sizes; $\mathbf{X}_0 = (x_1, x_2 \ldots, x_n) = (\min(\mathbf{S}_1), \ldots, \min(\mathbf{S}_n))$, Initialize iteration variable; i = 1.

Main step:

1. Initialize the selection set; $\mathbf{U} = \{1, 2, \ldots, n\}$; j=1.
2. Simulate the queueing model to obtain a vector $\mathbf{P}_{i,j}$ for all $j \in \mathbf{U}$. Fix x_t to the temporal optimal values where $t \neq j$. Then *f* becomes a function of one variable, x_j. Evaluate $f(x_j)$ for all j and all $x_j \in \mathbf{S}_j$. $\mathbf{P}_{i,j} = f(x_j)$, for $j \in \mathbf{U}$.
3. Calculate correlation coefficients, $C_{i,j} = \rho(\mathbf{P}_{i,j}, \mathbf{S}_j)$ where $j \in \mathbf{U}$.
4. Find the correlation coefficient $C_{i,k}$ with highest absolute value; $|C_{i,k}| \geq |C_{i,j}|$ for all $k \neq j$, $k \in \mathbf{U}$.
5. Update x_k in $\mathbf{X}_i$ to yield the best value of $\mathbf{P}_{i,k}$. If there are multiple optimal solutions, pick the smallest value of x_k, i.e., choose the smallest buffer size.
   If $\mathbf{P}_{i,k} = \mathbf{P}_{i-1,k}$, then no update.
6. Remove k from set U. If **U** is not empty, go to 2.
7. If $\| f(\mathbf{X}_i) - f(\mathbf{X}_{i-1}) \| < \varepsilon$, then stop. Otherwise, i = i + 1, and go to 1.

The local optimal solution is guaranteed. The main idea of this algorithm is that the buffer with the highest correlation to the performance is updated in each iteration and the performance is re-evaluated according to the change. The correlation helps to approach the optimal solution faster along the huge searching space of decision variables.

5 IMPLEMENTATION

Two simulation models were created to analyze traffic patterns for both independent and dependent (pipelined) processing units in a shared-memory system. These cases are complex and too difficult to analyze with mathematical methods. The simulation programs are developed with the aid of CSIM. CSIM is a process-oriented, discrete-event simulation package for use by C or C++ programs. In a CSIM model, a process represents a processing unit. A facility in CSIM is used to model the bus resource with a single queue. Processes operate in a simulated environment, controlled by the execution supervisor with respect to the passage of simulated time. For the independent processing model, each process issues bus request independently from other processes. The request is either predetermined or generated randomly by the statistics function. For the pipelined processing model, processing activities are synchronized and scheduled. The bus requests issues from two neighboring processing units are highly correlated. The structure and the flow of the design are shown in Figure 3.

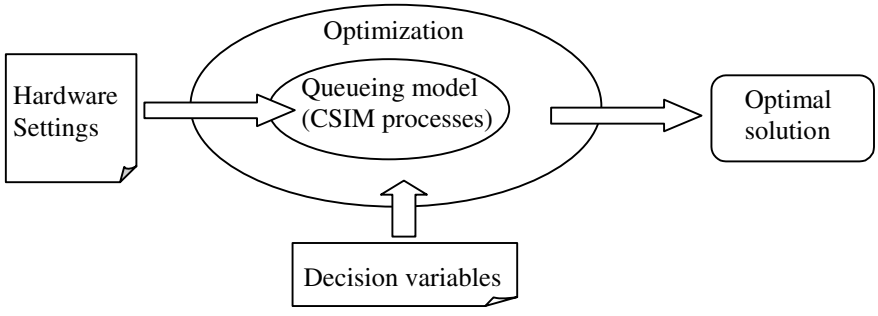

Figure 3: The Structure of the Design

6 SIMULATION EXPERIMENTS

The experiments are done for both independent PUs and pipelined PUs. The bus request patterns in the system are mixed-type, i.e., combining both deterministic and random types. The tests are to exploit the nature of system response and find the optimal setting for decision variables, I/O buffer sizes, with respect to the best system performance. The buffer size is divided into different levels, numbered from 1 to n, for the sake of displaying.

Table 1 and Figure 4 are experimental outcomes from a model with 5 independent PUs. Table 1 shows the optimal buffer solution with respect to different objective function, listed in the first column. In the example, buffer level 1, 2, 3, … and 8 stand for sizes 8, 16, 32, … and 1024 bytes. For instance, if the objective function is defined as the best throughput of PU3, then the solution is 1, 1, 8, 7, and 4 for buffer1, buffer2, … and buffer5, respectively. The interesting result is that each processing unit tends to make its buffer biggest to gain the best response, as shown in the diagonal line of the table. Figure 4 shows the relationship between total system waiting time and switching time over all possible configurations of buffers. Bus utilization in this example is 90.7%. As we can see, minimums of waiting time along the switching time axis linearly grows with switching time. This can be explained by taking micro views of the system. When bus utilization reaches the full state, any PU's memory access time, including switching time, is turned out to be other PUs' waiting time from their point of view. Thus, the waiting time increases with the switching time.

Table 1: Optimal Buffer Sizes for Different Objectives

Objective	**Buffer number**					**Optimal response (time unit/byte)**
	1	**2**	**3**	**4**	**5**	
System	7	8	8	6	5	0.233408
PU1	8	1	3	7	8	0.127541
PU2	3	8	2	6	7	0.131516
PU3	1	1	8	7	4	0.132472
PU4	5	1	1	8	8	0.127698
PU5	5	1	1	6	8	0.127989

For simplicity, the test model for pipelined PUs uses 3 processing units and 4 I/O buffers with 10 buffer levels for each. Figure 5 shows the results of all possible settings of 4 embedded buffers in the system. The vertical axis, bandwidth, represents the percentage of bus utilization. Each dot represents one outcome of simulation with respect to one setting. It illustrates the decision space and the multiple solutions to one given performance. Figure 6 shows the trace of searching algorithm for 5 iterations. The search started from different points, in this example, both cases converge into the same optimal solution. However, the algorithm cannot guarantee convergence to the same optimal

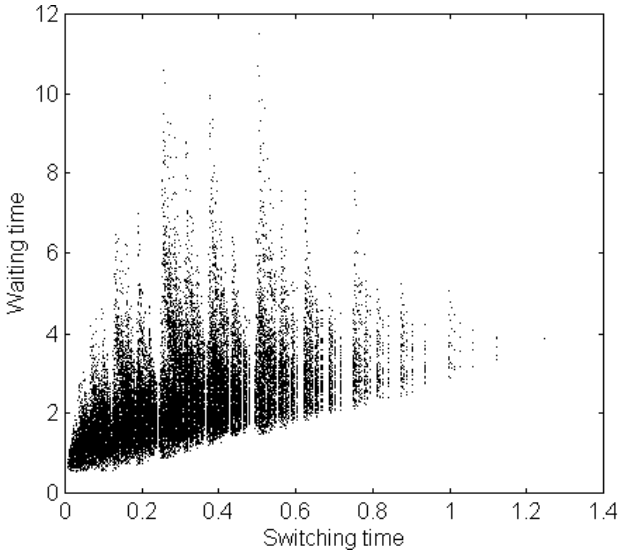

Figure 4: Switching Time vs. Waiting Time for 5 Independent PU model

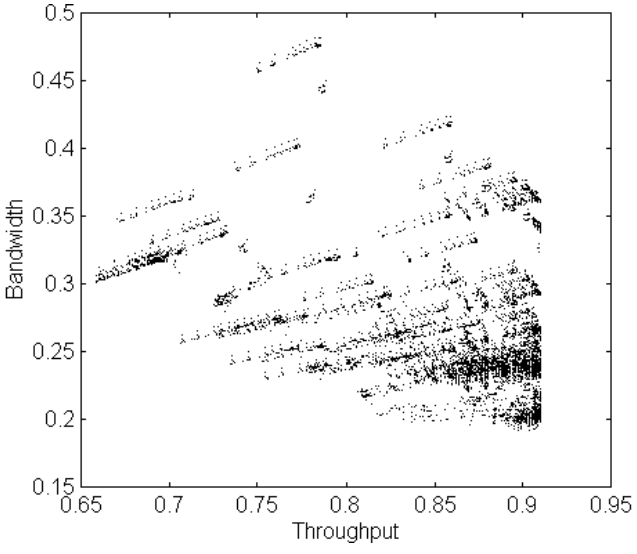

Figure 5: Decision Space for I/O Buffers

solution due to the existence of local optimal points. The local optimal solution is guaranteed.

7 CONCLUSION

The objective function in pipelined PU model is more specific than that in the independent PU model. From the hardware point, both physical structures are the same, whereas, from the bus controller, each processing unit is in different nature. In this paper, we focused on the effect of I/O components to system performance under the surveyed queueing models. A methodology was shown to define the problem, model the system and find the solution. An algorithm was developed to search for the optimal solution ef-

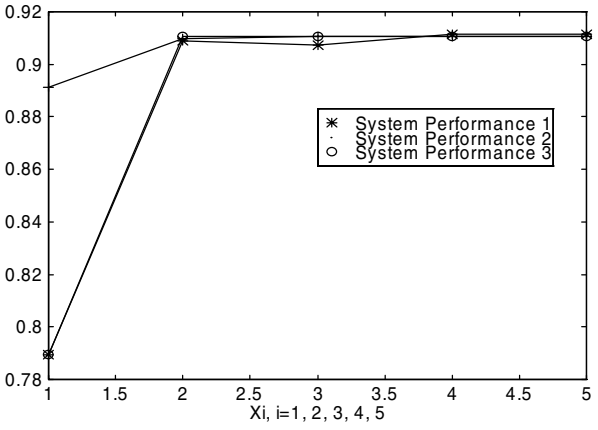

Figure 6: Trace of Performance

ficiently by exploiting the correlation of decision variables and the objective function. The optimal solution may not be unique. It does help the system designers to make proper decisions and optimize the design.

REFERENCES

Hillier, F. S., and G. J. Lieberman. 1995. Introduction to Operations Research, ISBN 0-07-841447-4, sixth ed. Stanford: McGraw Hill.

Jonkers, H. 1994. Queueing Models of Shared-Memory Parallel Applications. *Computer and Telecommunication Systems Performance Engineering.* London: Pentech Press.

Kornecki, and J. Zalewski. 1998. Simulation of Multiprocessor Bus Systems for Real-time Applications. *Proceedings of the 1998 Conference on Simulation Methods and Applications*, The Society for Computer Simulation, pp. 74-81. San Diego.

Lauzon, D., and A. Vincent, and L. Wang. 1996. Performance Evaluation of MPEG-2 Video Coding for HDTV. *IEEE transactions on Broadcasting*, Vol. 42, No. 2.

Li, J. H., and N. Ling. 1999. Architecture and Bus Arbitration Schemes for MPEG-2 Video Decoder. *IEEE Transactions on Video Technology*, Vol.9, No. 5, pp. 727-736. Santa Clara.

AUTHOR BIOGRAPHIES

J. H. LI received her B.S. degree in information and computer engineering from Chun-Yuan Christian University, Taiwan, in 1988, and her M.S. degree in computer engineering from Santa Clara University, California, U.S.A., in 1996. She worked as an engineer at Matsushita Elec. Ins. of Tech. Co. from 1988-1990. She has been a research assistant and teaching assistant at Santa Clara University. She is currently pursuing a Ph.D. degree in computer engineering at Santa Clara University. Her research interests include queueing theory, operations research, image compression, video technology, signal processing, VLSI design and parallel computing.

J. HOLLIDAY is an Assistant Professor at Santa Clara University. She received her B.S. at UC Berkeley and her M.S. at Northeastern University in Boston. Her Ph.D. is from UC Santa Barbara. Her professional interests include distributed replicated databases, distributed operating systems, and multicast, mobile, and ad-hoc networks.

G. FEGAN is an Associate Professor and Chair of the Applied Mathematics Department at Santa Clara University. He received his B.S. degree from the University of San Francisco, an M.A. from San Francisco State University, an M.A. from San Jose State University, and the Ph.D. from Oregon State University (1973). He has been a full-time faculty member at Santa Clara since 1987. He specializes in statistics.