PERFORMANCE CHARACTERISTICS FOR HIERARCHICALLY ORGANIZED MULTIPROCESSOR COMPUTER SYSTEMS WITH GENERALLY DISTRIBUTED PROCESSING TIMES

U. HERZOG

Institute of Mathematical Machine and Data Processing,
University of Erlangen-Nuremberg, Erlangen, Federal Republic of Germany

TO PROF. DR.-ING. A. LOTZE,

Director of the Institute of Switching Techniques and Data Processing, University of Stuttgart, in honor of his 65th birthday (7.2.1980).

He roused our interest in traffic theory. He showed us how to attack traffic problems, how to solve them and how to prepare the results for presentation and practical application. He continuously supported our professional progress and the contact between all members of his team and colleagues all over the world. Good luck and best health for the future!

ABSTRACT

Performance modeling and evaluation for hierarchically organized multiprocessor computer systems has to take into consideration the particulars of the hardware structure, the operating mode, and the structure of the application programs,as well. This is possible by a new class of queuing systems.

We briefly discuss the modeling technique and attack the problem of generally distributed processing times. For this we compare various solution techniques and show that the socalled piecewise solution is very attractive with regard to approximating measured data and numerical evaluation, as well.

1. INTRODUCTION

Multiprocessor computer systems with two or three processing units have been built since many years. Due to inexpensive hardware-components and mini- and microcomputers there is an increasing interest in building systems with some ten or even hundreds of processors [1-3].

Rather than running independent tasks on different processors one also tries to take advantage of the parallelism inherent in many problems, i.e. application programs are decomposed into sets of parallel cooperating subtasks and processed in parallel, when possible. So we may increase not only the throughput of a system: runtimes (and therefore response-times) for individual application programs may be reduced significantly, too. Then, however, difficult coordination problems (synchronization between tasks, data- and load-sharing, etc.) may occur.

In two recent papers [4,5] we surveyed the various problems and modelled the traffic flow by a new class of queuing systems. And we analyzed the various models mostly under Markovian assumptions.

In this paper we show for the most important models how to extend these results also for generally distributed processing times. First, we make some general remarks on hierarchically organized computer systems and discuss queuing models which capture the particulars of such systems. Secondly, we analyze these models and derive characteristic performance values, such as throughput, mean response time and distribution functions. Finally, we conclude with some remarks on ongoing research.

2. HIERARCHICALLY ORGANIZED COMPUTER SYSTEMS

2.1 General remarks

Hierarchical structures are transparent since we may distinguish clearly between organizational and application work: it is possible to concentrate coordination problems while distributing independent user tasks.

Typical examples are the EGPA-project [2], the multiprocessor system at the SUNY [3], the Siemens-system SMS, MOPPS, X-TREE and others.

Fig. 1 shows a basic two level hierarchy and a typical timing diagram:

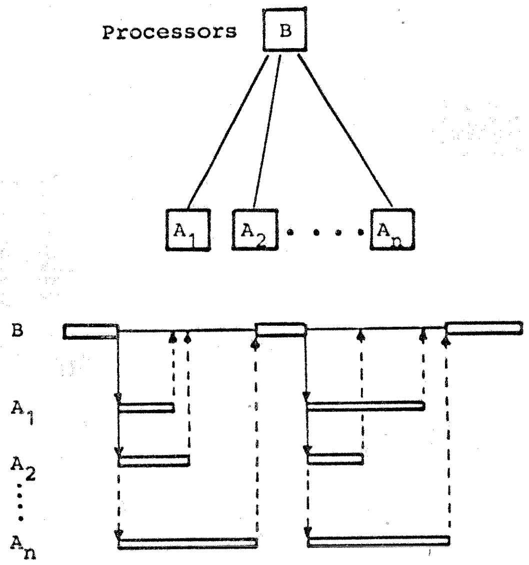


Fig. 1: Structure and timing diagram for a two level hierarchy, processor B for organization and coordination, processors A_i for application processing $(i=1,2,\dots,n)$.

- At first the source program is translated, loaded and then started by the B-processor.
- The B-processor then initiates the execution of n independent subtasks by the A-processors.
- Having completed its subtask, each A-processor has to inform the B-processor.
- Postprocessing and preparation of a new loop cycle by the B-processor is possible if and only if all subtasks are completed (synchroni-

zation).

Queuing models which allow to describe and analyze the traffic flow including the above synchronization problem are shown in figures 2 to 4.

2.2 Modeling

- Monoprogramming

For reason of simplicity and transparency we first assume monoprogramming for both B- and A-processors, cf. fig. 2.

- Newly arriving demands (source programs) are buffered in the input queue.
- If the "inner" system is empty the first demand waiting in the input queue is processed by the B-processor.
- The B-processor generates n independent sub-demands and distributes them simultaneously among all n A-servers (more sophisticated transfers, cf. section 4).
- After completion, each sub-demand is buffered in the corresponding synchronization queue of the B-server. If all n sub-demands are buffered, they are removed simultaneously (symbolized by the lying bracket ⊔) from the n parallel synchronization queues and processed in one step.
- After completion there are two possibilities: the (complete) demand leaves the system or n new sub-demands are generated simultaneously and a new loop-cycle is started.

The corresponding closed queuing model is shown in fig. 3.

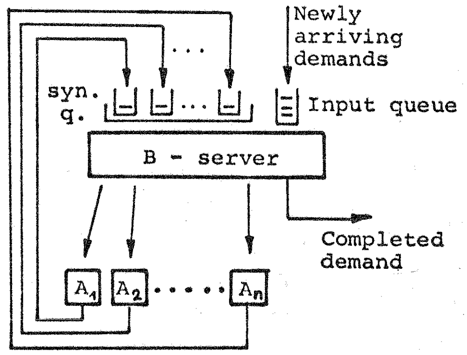


Fig. 2: Open model for monoprogramming

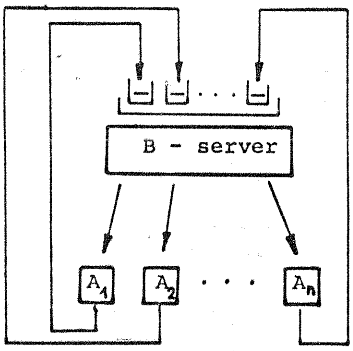


Fig. 3: Closed model for monoprogramming

- Multiprogramming

If we allow multiprogramming for both B- and A-processors several programs may be interleaved. In principle, the structure of the queuing models is the same as before. However, queues may build up in front of the A-processors, too.

- Mixed multi- and monoprogramming

Multiprogramming allows to increase system throughput. However, for reason of simplicity and transparency of the operating system there is a trend to introduce monoprogramming again. For many applications a mixture of both seems to be an efficient solution: multiprogramming for the B-processor and monoprogramming for the A-processors. Figure 4 shows the corresponding (closed) queuing model and is rather self-explanatory:

Be given a number m of independent demands $(t_1,\dots,t_i,\dots,t_m)$ to be served sequentially(!) by the B-server. After completion each task t_i $(i=1,\dots,m)$ generates n_i independent subtasks to be processed by the reserved A-processors A_{i1} to A_{in_i}. Task t_i may be started again if and only if all subtasks have been completed by the A-processors. If the B-server is busy, complete demands wait in front of the server and are served in the order of arrival (FIFO).

Note, synchronization is only necessary between sub-demands belonging together, an important fact for analysis.

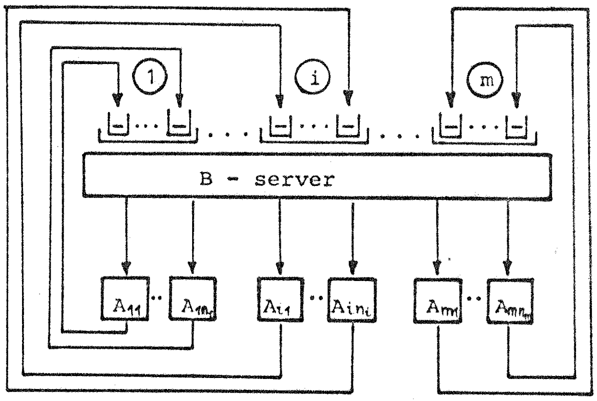


Fig. 4: Closed queuing model for mixed mode

Influence of application programs

Up to now we assumed that each application program to be processed is of that type shown in figure 1: a single serial program section is followed by several independent subsections which may be processed in parallel. If and only if these independent subtasks are completed a new cycle may start.

Problems are often of this type: algorithms for the solution of linear-algebraic or partial differential equation systems, optimization procedures, simulations including subruns for the purpose of estimating confidence intervals, problems of picture processing, etc. etc.

In analyzing the flow of information we assume this type of programs. Other program structures and the corresponding queuing models may be found in [4,5].

Although we have not investigated yet these other models under the new extended conditions, analysis seems to be straight forward.

Multi-level computer systems

Due to inexpensive hardware-components there is an increasing interest in building large systems with some ten, hundreds or thousands of processors. Then, hierarchically organized computer systems with three or more layers of computers may be the appropriate structure.

In modeling the traffic flow it is rather unwise and probably unsuccessful to describe and analyze the complicated flow of information by a single global queuing model.

An efficient and often successfully applied technique is the socalled "hierarchical modeling technique". For more details and examples, cf. also [4,5].

3. ANALYSIS

3.1 General remarks

As mentioned above, we concentrate on the analysis of monoprogramming-models. Recall, however, that for the mixed-mode-models synchronization is only necessary between sub-demands belonging together. So, if we are able to analyze the individual synchronization processes, as shown next, the mixed-mode problem can be reduced readily to the solution of a single-server system with finite population. In addition, multi-level models may be analyzed by the hierarchical modeling technique [4,5].

Traffic assumptions: Arrival processes (open model) are assumed to be Poissonian. From a didactic point of view only, it is reasonable to assume that each application program (demand) consists of a constant sequence of c cycles, cf. figure 5. The service requirement for each B-period (serial service) may be of general type with d.f. $F_B(t)$. Also, during the A-period (parallel sërvice), each individual subprocess may be arbitrarily distributed with d.f. $F_{Ai}(t)$, $i \in \{1,2,\ldots,n\}$.

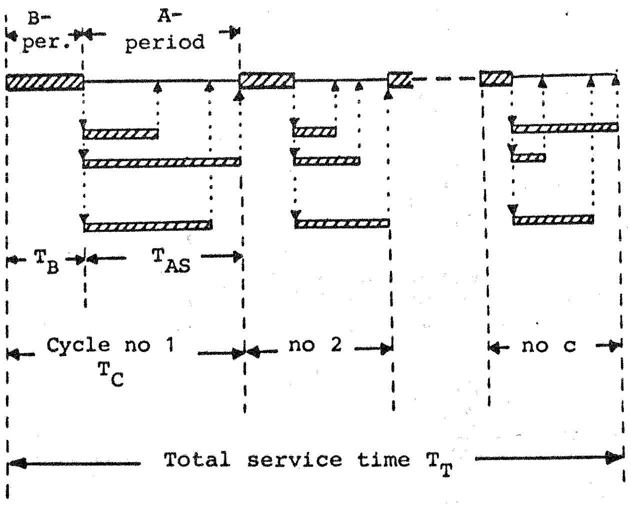


Fig. 5: Service requirements for a standard application program (c.f. text).

In analyzing the performance we assume stationarity and determine global performance values such as

- system throughput
- server utilization individually for each server
- mean numbers of A-servers working simultaneously
- mean number of demands and/or sub-demands waiting in front of servers (queue length)
- mean response time

Obviously, this is (in general) only possible if we investigate at first the microscopic behavior and determine

- mean, variance and d.f. of the A-period (often called synchronization time), i.e. the time interval between the moment "all parallel subprocesses start" and the event "the last parallel subprocess is completed".
- mean, variance and d.f. of the cycle time, the sum of the B- and A-period.
- mean, variance and d.f. of the total service time requirement for each application program.

We now start by investigating the microscopic behavior.

3.2 Microscopic behavior and general service times

In case of generally distributed service intervals there are at least three possibilities to determine moments and the d.f. of the time intervals we are interested in:

- The direct solution, i.e. applying fundamentals of probability theory we may derive these values directly from the given d.f.s of general type.
- The phase-type solution, i.e. given d.f.s are approximated by phase-type distributions and a multidimensional Markov-analysis is made.
- The piecewise solution, i.e. given d.f.s are approximated by piecewise exponentials and again we take advantage of the special properties of the exponential d.f.

We survey all three possibilities and discuss their advantages and disadvantages by means of examples.

3.2.1 The direct solution

Be given one demand D, the total service time T_T of which consists of a sequence of serial and parallel executable service requirements as shown in figure 5.

Obviously, the A-period ("synchronization time") is determined by the maximum of n service times T_{Ai}, $i \in \{1,2,\ldots,n\}$. Therefore, its d.f. $F_{AS}(t)$ is given by the product of all d.f.s $F_{Ai}(t)$:

$$F_{AS}(t) = P(T_S \leq t) = P(T_{A_1} \leq t) \cdot \ldots\ldots \cdot P(T_{A_n} \leq t)$$

$$F_{AS}(t) = \prod_{i=1}^{n} F_{A_i}(t)$$

Example: Suppose, all service intervals T_{Ai} are exponentially distributed with uniform service rates $\varepsilon_i = \varepsilon$. We then obtain

$$F_{AS}(t) = \prod_{i=1}^{n} F_{A_i}(t) = (1 - e^{-\varepsilon t})^n$$

$$= \sum_{k=0}^{n} \binom{n}{k} (-1)^k \cdot e^{-k\varepsilon t}$$

with mean and variance

$$E\left[T_{AS}\right] = \frac{1}{\varepsilon}\sum_{k=1}^{n}\frac{1}{k}$$

$$VAR\left[T_{AS}\right] = \frac{1}{\varepsilon^2}\sum_{k=1}^{n}\frac{1}{k^2} \quad \blacksquare$$

Since the cycle time T_C is the sum of the B-period (T_B) and the A-period (T_{AS}), its d.f. is determined by the convolution of these two d.f.s.

$$F_C(t) = F_B(t) * F_{AS}(t)$$

Example: Suppose the B-period is constant ($T_B = t_b$) and, as before all intervals T_{Ai} are exponentially distributed. Then

$$F_C(t) = \begin{cases} 0 & 0 \leq t < t_B \\ \sum_{k=0}^{n} \binom{n}{k}(-1)^k \cdot e^{-k\varepsilon(t-t_B)} & t_B \leq t < \infty \end{cases}$$

with mean and variance

$$E\left[T_C\right] = t_B + \frac{1}{\varepsilon}\sum_{k=1}^{n}\frac{1}{k}$$

$$VAR\left[T_C\right] = \frac{1}{\varepsilon^2}\sum_{k=1}^{n}\frac{1}{k^2} \quad \blacksquare$$

Finally, the d.f. of the total service time is determined by the c-fold convolution of the cycle time

$$F_T(t) = F_C(t) * F_C(t) * \ldots\ldots * F_C(t)$$

with mean and variance

$$E\left[T_T\right] = c \cdot E\left[T_C\right] = c \cdot E\left[T_{AS}\right] + c \cdot E\left[T_B\right]$$

$$VAR\left[T_T\right] = c \cdot VAR\left[T_C\right] = c \cdot VAR\left[T_{AS}\right] + c \cdot VAR\left[T_B\right]$$

This brief survey already shows that the direct solution is, in principle, very powerful. Hoffmann [6,7] studied this method in detail. In particular, if service intervals are of exponential or of phase type, explicit results may be obtained. In case of arbitrarily distributed time intervals, however, the evaluation of products and convolutions of d.f.s may cause trouble.

3.2.2 The phase-type solution

One of the most successful concepts in queuing theory is based on the idea to represent arbitrarily distributed time intervals by sums of convolutions of exponentially distributed random variables, the socalled "phases".

This concept was introduced by Erlang and generalized by Jensen and Cox (for more details, in particular for details about the efficient approximation of given d.f.s, cf. [8]). Analysis is done by multidimensional Markovian technique.

Example: Suppose, the d.f. of the B-period can be approximated by a hyperexponential d.f. with two phases:

$$F_B(t) = \sum_{j=1}^{2} q_j\left(1 - e^{-t/\mu_j}\right)$$

with $q_1 + q_2 = 1$. Furthermore, suppose that there are two service intervals T_{Ai}, $i \in (1,2)$, to be started in parallel, the d.f. of both may be approximated by an Erlang-distribution of order k.

$$F_{A_i}(t) = 1 - e^{-k\varepsilon t}\sum_{\xi=0}^{k-1}(k\varepsilon t)^{\xi}/\xi!$$

We now apply the standard technique well known for Markovian processes [9]. Figure 6 shows the state transition diagram from which we usually start the analysis.

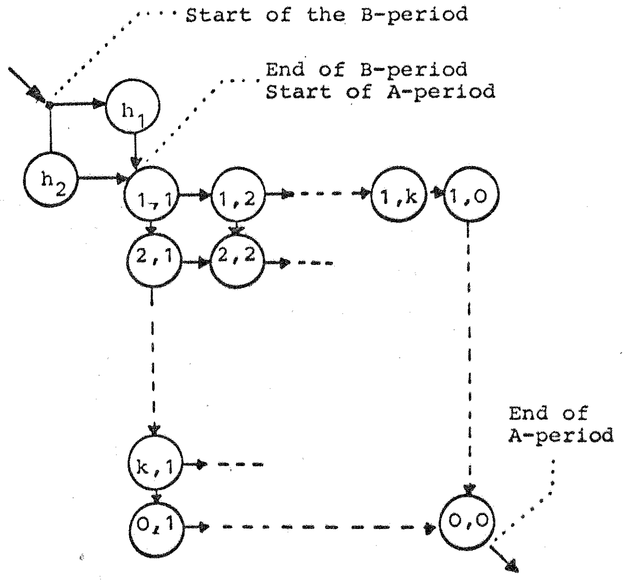


Fig. 6: State-transition diagram for the phase-type-solution, example (cf. text).

Obviously, mean and variance of the B-period are determined by the moments of the hyperexponential d.f.:

$$E\left[T_B\right] = \sum_{j=1}^{2}\frac{q_j}{\mu_j}$$

$$VAR\left[T_B\right] = 2 \cdot \sum_{j=1}^{2}\frac{q_j}{\mu_j^2} - E\left[T_B\right]^2$$

Boguslavsky-Gelenbe [10] and Hoffmann [11] investigated the state probabilities for the A-period from which the moments may be derived readily:

$$E\left[T_{AS}\right] = \frac{1}{k \cdot \varepsilon}\sum_{j=1}^{2}\binom{2}{j}(-1)^{j-1}\sum_{\xi_1=0}^{k-1}\sum_{\xi_j=0}^{k-1}\frac{\left(\sum_{\eta=1}^{j}\xi_\eta\right)!}{\prod_{\eta=1}^{j}\xi_\eta! \cdot j^{1+\sum_{\eta=1}^{j}\xi_\eta}}$$

$$VAR\left[T_{AS}\right] = \frac{1}{(k\varepsilon)^2} \cdot \left[2 \cdot \sum_{j=1}^{2}\binom{2}{j}(-1)^{j-1}\sum_{\xi_1=0}^{k-1}\sum_{\xi_j=0}^{k-1}\frac{\left(1 + \sum_{\eta=1}^{j}\xi_\eta\right)!}{\prod_{\eta=1}^{j}\xi_\eta! \cdot j^{2+\sum_{\eta=1}^{j}\xi_\eta}}\right] - E\left[T_S\right]^2 \quad \blacksquare$$

Mean and variance of the cycle and total service time distribution follow immediately by adding the corresponding partial results.

Again, we studied the phase-type method and found many interesting results [12,4,7]. If there are more than two parallel subprocesses and phase-type d.f.s of high order, however, a considerable amount of evaluation work may be necessary.

3.2.3 The piecewise solution

- Approximation technique

In order to represent generally distributed time intervals we can use, as the phase-concept does, exponential phases. Now, however, we devide these time intervals in different sections ("pieces") and approximate the shape of the actual d.f. within each "piece" by means of one exponential phase. Hence, the d.f. of the piecewise exponential of r-th order is given by [13]:

$$P(T\le t) = 1-\exp\left[-\sum_{\xi=1}^{j} t_{g_\xi}(\varepsilon_\xi-\varepsilon_{\xi+1})-\varepsilon_{j+1}t\right]$$

for

$$t_{g_j} < t \le t_{g_{j+1}}\ ,\ j=0,1,\dots,r,$$

$$0 = t_{g_0} < t_{g_1} < \dots < t_{g_r} < t_{g_{r+1}} \le \infty$$

$$0 \le \varepsilon_\xi \le \infty \quad (!)$$

the moments of which may be determined efficiently by a recursive technique [14].

Note that we include the extreme exponentials with zero and infinite service rate! We therefore include step-functions in our concept and are very flexible in approximating d.f.s of real measured time intervals. Figure 7a shows an example with three exponential pieces and a final step to unity. Figures 7b and 7c show two more examples demonstrating the flexibility of this method.

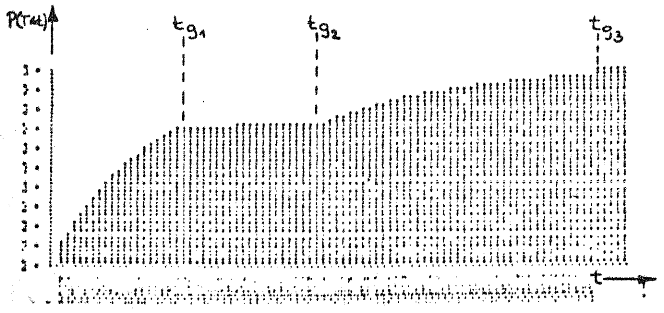
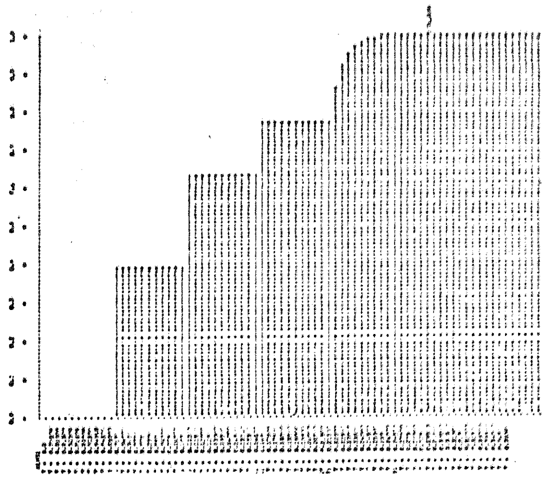


Fig. 7a: Approximation of generally distributed time intervals by exponential pieces.

Fig. 7b: Approximation of generally distributed time intervals by step functions and exponential pieces.
(Fig. see right column above.)



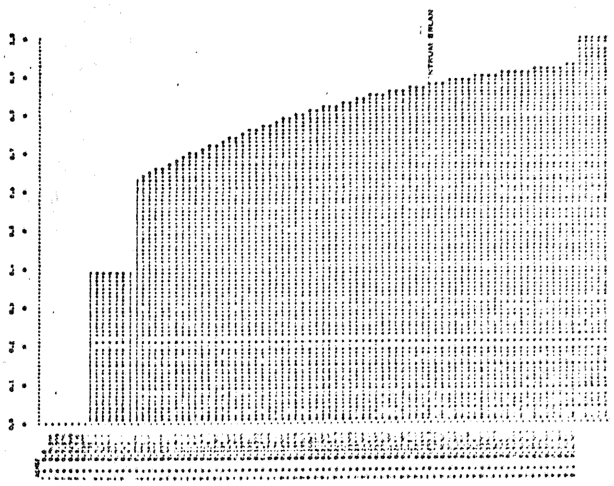


Fig. 7c: Third example for the piecewise fitting of generally distributed time intervals.

- Analysis

In order to determine the length of the A-period we may follow the same arguments as in section 3.2.1 and get directly the d.f.:

$$F_{AS}(t) = P(T_{AS}\le t) = \prod_{i=1}^{n} F_{Ai}(t)$$

For the most important case of identical d.f.s $F_{Ai}(t)$ we obtain directly

$$F_{AS}(t) = (1-c_j\cdot e^{-\varepsilon_{j+1}t})^n$$

$$= \sum_{p=0}^{n}\binom{n}{p}(-1)^p\cdot c_j^p e^{-\varepsilon_{j+1}pt}$$

$$\text{for } t_{g_j}\le t \le t_{g_{j+1}}$$

$$j=0,1,\dots\dots,r$$

$$\text{with} \quad c_j = e^{-\sum_{\xi=1}^{j} t_{g_\xi}(\varepsilon_\xi-\varepsilon_{\xi+1})}$$

Interesting moments, such as mean and variance can be derived easily:

$$E[T_{AS}] = \sum_{j=0}^{r} \int_{tg_j}^{tg_{j+1}} t \cdot d\,F_{AS}(t)$$

$$= \sum_{p=1}^{n} \binom{m}{p} (-1)^{p+1} \sum_{j=0}^{r} \frac{c_j^{\,p}}{\varepsilon_{j+1} p} \cdot \Big[-e^{-x}(x+1) \Big]_{x=p\varepsilon_{j+1}tg_j}^{x=p\varepsilon_{j+1}tg_{j+1}}$$

$$VAR[T_{AS}] = \sum_{p=1}^{n} \binom{m}{p} (-1)^{p+1} \cdot \sum_{j=0}^{r} \frac{c_j^{\,p}}{\varepsilon_{j+1}^{2} \cdot p^{2}} \cdot$$

$$\cdot \Big[-e^{-x}(x^2+2x+2) \Big]_{x=p\varepsilon_{j+1}tg_j}^{x=p\varepsilon_{j+1}tg_{j+1}} - E[T_{AS}]^2$$

Cycle time and total service time may be obtained accordingly.

Example: Be given a piecewise exponential d.f. of first order for each parallel subprocess ($i \in \{1,2,\ldots,n\}$):

$$F_{A_i}(t) = \begin{cases} 1 - e^{-\varepsilon_1 t} & 0 \le t \le tg_1 \\ 1 - e^{-tg_1(\varepsilon_1-\varepsilon_2)} \cdot e^{-\varepsilon_2 t} & tg_1 \le t < \infty \end{cases}$$

Then, the d.f. of the A-phase is given by

$$F_{AS}(t) = \begin{cases} \sum_{p=0}^{n} \binom{n}{p} (-1)^p \, e^{-\varepsilon_1 p t} & 0 \le t \le tg_1 \\ \sum_{p=0}^{n} \binom{n}{p} (-1)^p \, e^{-tg_1(\varepsilon_1-\varepsilon_2)p} \cdot e^{-\varepsilon_2 p t}, & tg_1 \le t < \infty \end{cases}$$

furthermore

$$E[T_{AS}] = \sum_{p=1}^{n} \frac{1}{p} \binom{m}{p} (-1)^{p+1} \left\{ \frac{1}{\varepsilon_1}\left(1 - e^{-\varepsilon_1 p tg_1}\right) + \frac{1}{\varepsilon_2} \cdot e^{-\varepsilon_1 p tg_1} \right\}$$

$$VAR[T_{AS}] = \sum_{p=1}^{n} \frac{1}{p^2} \binom{m}{p} (-1)^{p+1} \left\{ \frac{2}{\varepsilon_1^2} \left[1 - e^{-\varepsilon_1 p tg_1}(\varepsilon_1 p tg_1 + 1) \right] + \right.$$

$$\left. + \frac{2}{\varepsilon_2^2} \cdot e^{-\varepsilon_1 p tg_1} (\varepsilon_2 p tg_1 + 1) \right\} - E[T_S]^2 \quad ■$$

Again, mean and variance of the cycle time and the total service time distribution follow immediately by adding the corresponding partial results.

Conclusion

The advantage of the piecewise solution is twofold:

1) It is fairly easy to approximate real measured time intervals by means of piecewise exponentials especially if we include step functions.
2) The expressions for d.f.s as well as moments are rather elementary and easy to evaluate, even if there is a large number of parallel subprocesses. This advantage may vanish to some extend if there are many subprocesses of different type, say more than ten. Note, however, that subprocesses of the same type are most important for practical application since they guarantee a balanced usage of all resources.

3.3 Global performance values

- Closed model, fig. 3

Be given the mean length of the B-period $E[T_B]$, the d.f. $F_{Ai}(t)$ and its mean $E[T_{Ai}]$ for each parallel subprocess, $i \in \{1,2,\ldots,n\}$. Then, the mean length of the A-period $E[T_{AS}]$, the mean cycle time $E[T_c]$ and the mean total service time $E[T_T]$ may be determined as shown before. Obviously, the utilization of the top processor is

$$Y_B = \frac{E[T_B]}{E[T_c]}$$

Utilization of the A-processors

$$Y_{A_i} = \frac{E[T_{Ai}]}{E[T_c]} \qquad i \in \{1,2,\ldots,n\}$$

Throughput

$$\rho = \frac{1}{E[T_T]}$$

Mean number of A-processors working simultaneously during the A-phase

$$E[I] = \frac{\sum_{i=1}^{n} E[T_{Ai}]}{E[T_{AS}]}$$

- Open model, fig. 2

External arrivals are assumed to be Poissonian with arrival rate λ. Mean, variance and d.f. of the total service time for each demand can be determined according to section 3.2. Therefore, the analysis of global performance characteristics is possible just by using well known results from M/G/1-analysis [9].

Mean number of demands, waiting in the input queue

$$E[X] = \frac{\lambda^2 \cdot E[T_T]^2}{2(1-\lambda \cdot E[T_T])} \cdot \left\{ 1 + \frac{VAR[T_T]}{E[T_T]^2} \right\}$$

Mean response time

$$E[T_R] = \frac{\lambda \cdot E[T_T]^2}{2(1-\lambda E[T_T])} \cdot \left\{ 1 + \frac{VAR[T_T]}{E[T_T]^2} \right\} + E[T_T]$$

Utilization of the top processor

$$Y_B = \lambda \cdot E[T_T] \cdot \frac{E[T_B]}{E[T_c]}$$

Utilization of the A-processors

$$Y_{Ai} = \lambda \cdot E[T_T] \cdot \frac{E[T_{Ai}]}{E[T_c]}$$

Mean number of A-processors working simultaneously (during the A-phase):

$$E[I] = \frac{\sum_{i=1}^{n} E[T_{Ai}]}{E[T_{AS}]}$$

Mean number of A-processors working simultaneously (in total):

$$E[I]^* = \lambda \cdot E[T_T] \quad \frac{\sum_{i=1}^{n} E[T_{Ai}]}{E[T_c]}$$

4. SUMMARY

The intention of this paper is to demonstrate that there are various possibilities to model and to analyze the traffic flow in hierarchically organized multiprocessor computer systems.

Modeling general distribution functions (d.f.s) by the method of phases is a well known and very successful technique. For systems with parallel subprocesses, however, the piecewise solution is an alternative which may help to reduce the complexity of the modeling and evaluation technique.

We presented solution techniques which allow to analyze hierarchically organized computer systems with two levels of processing and monoprogramming. Recall, however, that these fundamental results allow to model and to analyze also multi-level hierarchies and mixed multi- and monoprogramming operating mode.

There are still many interesting problems. E.g., we have to automatize our interactive program for fitting measured data by means of piecewise exponentials and step functions. Other important problems, we are working at, are refinements of our queuing models: We may include transfer times for code and data, signalling overhead, priorities, etc. etc. And we have to search for solution techniques which may be particularily advantageous for our specific problems.

REFERENCES

[1] FULLER, S. et al.: Multi-Microprocessors: An Overview and Working Example. Proc. IEEE Vol. 66, No. 2 (1978), pp. 216-228.

[2] HÄNDLER, W.; HOFMANN, F.; SCHNEIDER, H.J.: A General Purpose Array with a Broad Spectrum of Applications. In: Computer Architecture (Händler ed.), Informatik-Fachberichte 4, Springer-Verlag (1975), pp. 311-335.

[3] HARRIS, J.A.; SMITH, D.R.: Hierarchical Multiprocessor Organizations. Computer Architecture News, Vol. 5, No. 7 (1977), pp. 41-48.

[4] HERZOG, U.; HOFFMANN, W.: Synchronization Problems in Hierarchically Organized Multiprocessor Computer Systems. Proceedings of the 4th International Symposium on Modeling and Performance Evaluation of Computer Systems, Feb. 6-8, 1979, Techn. University, Vienna. To be published by North Holland.

[5] HERZOG, U.; HOFFMANN, W.; KLEINÖDER, W.: Performance Modeling and Evaluation for Hierarchically Organized Multiprocessor Computer Systems. Paper submitted to Intern. Conf. on Parallel Processing, Bellaire, Michigan, August 21-24, 1979.

[6] HOFFMANN, W.: Verteilungsfunktion der Synchronisationszeit T_S für das geschlossene Synchronisationsmodell mit beliebiger Anzahl von A-Prozessoren und einem Gesamtauftrag. Unpublished memo, Dec. 20, 1978.

[7] HOFFMANN, W.; KLEINÖDER, W.: Warteschlangenmodelle für Mehrprozessorsysteme: maximale Bedienungszeit von l parallel arbeitenden Bedienungsstationen. Submitted to GI-Jahrestagung, Bonn, October 1-5, 1979.

[8] BUX, W.; HERZOG, U.: The Phase Concept: Approximation of Measured Data and Performance Analysis. Proc. IFIP-Symp. on Comp. Performance Modeling, Measurement and Evaluation, Eds. Chandy and Reiser, North Holland (1977), pp. 23-38.

[9] KLEINROCK, L.: Queuing Systems, Vol. 1: Theory, John Wiley and Sons, 1975.

[10] BOGUSLASKI, L.; GELENBE, E.: A Communication Protocol and a Problem of Coupled Queues. Proc. 4th Intern. Symp. on Modeling and Performance Evaluation of Computer Systems, Feb. 6-8, 1979, Techn. University, Vienna. To be published by North Holland.

[11] HOFFMANN, W.: Das einfache Synchronisationsmodell mit Erlang-k-verteilten Bedienungszeiten und l = 2 A-Prozessoren. Unpublished memo, March 21, 1979.

[12] HOFFMANN, W.: Warteschlangenmodelle für die Parallelverarbeitung. Ph.-D.-Thesis, University of Erlangen-Nürnberg, 1978.

[13] WALKE, B.: Realzeitrechner-Modelle, Theorie und Anwendung, Oldenbourg Verlag, München-Wien, 1978.

[14] NOETH, E.; HERZOG, U.: The piecewise exponential distribution function. Unpublished memo, March-April 1979.