Systems Modeling and Performance Evaluation
H.D. Schwetman Editor

Analysis of Locking Policies in Database Management Systems

D. Potier and Ph. Leblanc
INRIA, France

Consistency control has to be enforced in database management systems (DBMS) where several transactions may concurrently access the database. This control is usually achieved by dividing the database into locking units or granules, and by specifying a locking policy which ensures integrity of the information. However, a drawback of integrity enforcement through locking policies is the degradation of the global system performance. This is mainly due to the restriction imposed by the locking policies to the access of transactions to the database, and to the overheads involved with the management of locks. A framework for the quantitative analysis of the impact of these factors on the performance of DBMS is presented in this paper. In a first step, the main factors which determine the behavior of these systems are pointed out and analyzed independently. The results hereby obtained are aggregated in a second step to yield a global performance evaluation. Throughout this hierarchical modeling approach various analytical techniques are used and the results are illustrated by numerical examples. The paper concludes by pointing out the final results' sensitivity to some basic assumptions concerning transaction behavior and the need for more experimental studies in this area.

Key Words and Phrases: consistency, concurrency, database management, performance modeling, queueing models, queueing networks

CR Categories: 3.50, 4.33, 8.1

Authors' present address: D. Potier and Ph. Leblanc, INRIA, Domaine de Voluceau, Roquencourt, B.P. 105, 78150 Chesnay, France.

1. Introduction

Quantitative analysis of locking mechanisms and of their impact on the performance of transaction systems have received relatively little attention [2, 9, 10]. Although numerous concurrency mechanisms have been proposed and implemented, there is an obvious lack of experimental as well as analytical studies of their behavior and their influence on system performance. The present state of the art in this area can be compared to the situation in the late '60s when the first virtual memory systems were being implemented: At that time the now classical concepts of program behavior (locality of references, working-sets, memory management policy, etc.) were only emerging and it took several years to fully understand and master them.

In order to perform a quantitative analysis of transaction systems of the performance, it is essential to point out the factors that determine the behavior of these systems. Three main factors may be identified:

1. *Transaction Behavior.* It may be described by the pattern of references of the transactions to the subsets (logical or physical) of the database and by the way the central processing unit (CPU) and input/output (I/O) resources of the system are used by the transactions. Given a locking mechanism, the reference pattern is obviously a determining factor: Depending on the way references are distributed over the database, the locking mechanisms will have a more or less drastic effect.

However, as noted above, little experimental evidence on reference patterns of transactions is available. In particular, the existence of properties of sequentiality or locality in the access pattern is still a controversial issue [11, 12, 13, 15]. Faced with this situation, any analysis relies upon rather strong assumptions. For instance, in the analysis of locking granularity presented in [9], Ries and Stonebraker characterize the behavior of a transaction by the number of entities referenced, and assume that the number of granules locked by a transaction is proportional to the number of entities referenced.

In our analysis, we shall use a somewhat more sophisticated probabilistic model of transaction behavior although it also makes a strong assumption on the distribution of references over the database.

2. *Locking Scheme.* Locking schemes may be physical or logical depending on whether the units of locking are physical or logical subsets of the databases. We restrict our attention in this paper to physical locking. A physical locking mechanism is characterized by three main factors:

(a) The Lock Acquisition Policy (LAP) which specifies at which instant of the transaction lifetime locks are requested. LAPs are basically split into two categories: static LAPs (SLAP) and dynamic LAPs (DLAP). Under SLAP all the locks needed for executing a transaction are requested at the initiation of the transaction. The

transaction is executed only if all its requested locks are granted; otherwise it is blocked. Under a DLAP, locks are requested by a transaction on demand, whenever a transaction references a granule that is not already locked.

(b) The Lock Release Policy (LRP) which specifies when the locks acquired by a transaction are released. In order to maintain the consistency of the database a static LRP (SLRP) has to be used, i.e., a LRP where a transaction releases its locks only when its execution has been completed.

Following [9], the analysis presented in this paper is carried out by assuming that the locking policies used are SLAP and SLRP. The main advantages of this combination of LAP and LRP is that deadlocks are avoided and that, from the point of view of the analyst, its study is easier. However, it should be kept in mind that locks' mean lifetimes are longer under a SLAP than under a DLAP, and hence that a SLAP restricts the transaction's accesses to the database more severely than a DLAP.

(c) The size of the physical locking units or granules (assuming that all locking units have the same size). Given a LAP and a LRP, the remaining design parameter is the granule size. Obviously, in theory, the smaller the granules, the smaller the constraints imposed by the locking mechanism. However, at this point, secondary factors such as locking overheads have to be taken into account, since their influence may become important when the number of granules is large.

3. *The Multiprogramming Environment.* The multiprogramming environment describes the resources (CPU, I/O) that the transaction system uses for the actual processing of transactions. These resources, their characteristics and organization, have an important impact on the global performance. Of the three factors we have identified, the latter is also presently the best understood and mastered. Within the past ten years much experimental and analytical effort has been devoted to this area [4, 6], and the appropriate methods and tools for the analysis of the multiprogramming environment do exist.

Once these factors have been identified, we must select an approach to analyze them and quantify their influence. In [7, 9] simulation models are used. Although this technique has the advantage of imposing no restriction on the complexity of the analyzed mechanisms, it cannot always provide clear insight into and an understanding of the system performance and the role of key assumptions.

Our approach in this paper is based on hierarchical analytical modeling [3]. The basic idea underlying this approach is that in a first step each critical factor should be analyzed independently, clearly stating assumptions and their consequences; the results hereby obtained are aggregated in a second step to yield a global performance evaluation and final conclusions.

A broad presentation of this approach is given in Section 2. Section 3 is devoted to the model of transaction behavior and the analysis of the locking mechanism. The multiprogramming environment is studied in Section 4. Final results on the global performance are presented and discussed in Section 5. Conclusions are given in Section 6.

Fig. 1.

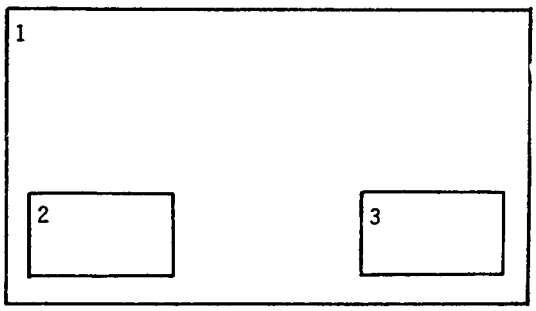
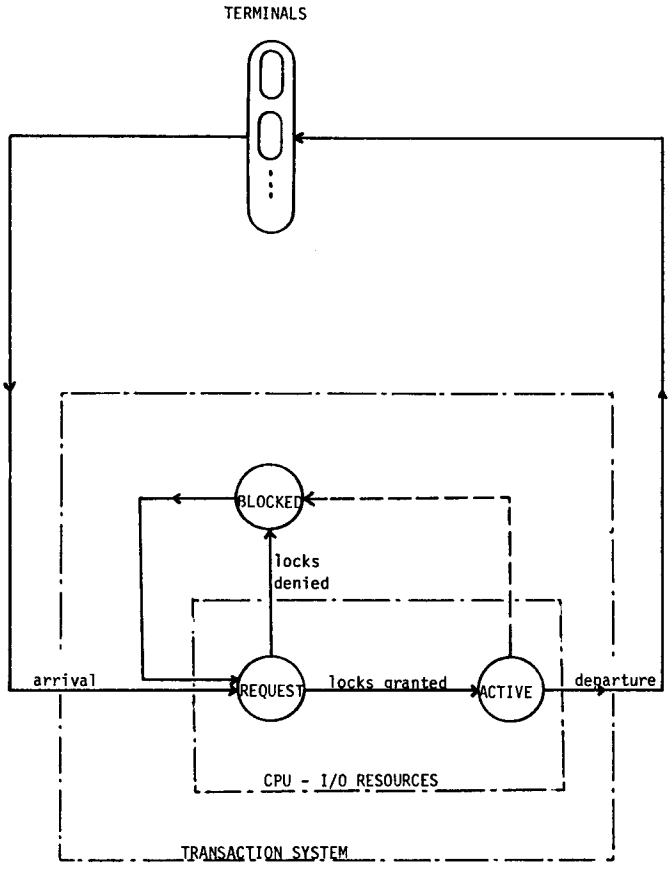


Fig. 2.



2. Presentation of the Approach

The transaction system is analyzed using a hierarchical modeling approach. Three levels of modeling are considered, as represented in Figure 1.

At level 1, the different stages transactions go through during their lifetimes are described. The model used at this level is represented in Figure 2. Transactions are

issued from a set of interactive terminals. Upon arrival of a new transaction in the system, all the locks needed for the complete execution of the transaction are requested. Lock requests are processed by the CPU and I/O resources of the system. During the "locks request" phase the transaction is in the state REQUEST. Upon completion of this phase, locks requested are either granted or denied. If locks are granted, the transaction enters the state ACTIVE and it is executed by the CPU and I/O resources of the system until completion. It then returns to the terminal that has issued it. If locks are denied, the transaction is sent into the BLOCKED queue and enters the state BLOCKED.

BLOCKED transactions are removed from the BLOCKED queue in order to enter a new "locks request" phase when an ACTIVE transaction leaves the system and releases its locks. The maximum number of BLOCKED transactions removed from the BLOCKED queue at each departure is a control parameter of the transactions' scheduling policy.

The organization and operations of the CPU and I/O resources of the system are described and analyzed at level 2. These resources are used by the transactions when they are either requesting locks (state REQUEST) or being executed (state ACTIVE). The analysis of level 2 will be conducted to derive the processing rates of ACTIVE transactions and REQUEST transactions depending on the number of these transactions sharing the CPU and I/O resources.

The behavior of transactions during their locks request phase will be analyzed at modeling level 3. Given a simple model of the transaction references to the entities of the database and the number of granules in the database, we will derive the probability distribution of the number of blocked granules depending on the number of ACTIVE transactions. These results will be used to obtain the probability that locks are granted to a transaction at the end of its locks request phase depending on the number of ACTIVE transactions.

The three levels of modeling will be presented and analyzed in the following order: level 3, level 2, and level 1.

3. Analysis of Transaction Behavior and Locking Mechanisms

3.1 The Physical Database

The physical database consists of cells that represent the elementary physical unit of storage (this will be, for example, a disk sector) and it is divided into m granules, where the granule is the locking unit.

3.2 Transaction Behavior

The behavior of a transaction consists of a sequence of accesses to the cells of the database. This sequence is defined by the following assumptions:

(1) a transaction makes n accesses during its execution;
(2) accesses are uniformly distributed over the cells of the database;
(3) accesses are independent.

Given these assumptions we shall derive the probability $P_k(l)$, $l = 1, \ldots, m$, that l granules are locked when k transactions are simultaneously active, and the probability q_k that a new transaction enters the set of active transactions when k transactions are already active. These results will then be illustrated by numerical examples.

3.3 Derivation of $P_k(l)$ and q_k

Let $x(p)$, $p = 1, \ldots, m$ be the probability that the number of distinct granules accessed by a transaction is p, given that the transaction makes n accesses. The derivation of $x(p)$ is similar to a standard combinatoric problem: given n like objects and m unlike buckets, what is the probability of leaving p buckets nonempty after having uniformly distributed the objects into the m buckets? Applying standard results of combinatorics (see for instance [8, pp. 103]), we then have

$$x(p) = \binom{m}{p}\binom{n-1}{p-1}\Big/\binom{m+n-1}{n} \tag{1}$$

and, obviously,

$$P_1(l) = x(l),\ l = 1, \ldots, m. \tag{2}$$

In order to derive $P_k(l)$, we write the following recurrence formula between $P_k(l)$ and $P_{k-1}(l)$,

$$P_k(l) = \frac{1}{\sum_{j=1}^{m-1} P_{k-1}(j)} \sum_{i=1}^{m-1} P_{k-1}(i) \cdot \frac{X_i(l-i)}{\sum_{j=1}^{m-i} X_i(j)} \tag{3}$$

where

$$\begin{aligned} X_i(p) &= x(p)\binom{m-i}{p}\Big/\binom{m}{p} \\ &= \binom{m-i}{p}\binom{n-1}{p-1}\Big/\binom{m+n-1}{n}. \end{aligned} \tag{4}$$

In eq. (4), $X_i(p)$ is defined as the probability that a transaction is granted p granules given that i granules are already locked. The terms in the denominator of eq. (3) are normalization factors: Since the kth transaction can become active only if there are less than m granules locked by the $(k-1)$st transactions, the probabilities $p_{k-1}(i)$, $i = 1, \ldots, m-1$ are normalized by the sum $\sum_{j=1}^{m-1} p_{k-1}(j)$; in the same fashion, given i, the maximum number of new granules locked by the kth transaction is $m - i$, so that the probabilities $X_i(p)$ in eq. (3) are weighted by $\sum_{j=1}^{m-i} X_i(j)$.

From $P_k(l)$, we can derive the mean number $\bar{l}_k$ of granules locked when k transactions are active

$$\bar{l}_k = \sum_{l=1}^{m} i \cdot P_k(i) \tag{5}$$

and we straightforwardly obtain the probability q_k that

Table I. q_k.

n	m	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
2	2	.222	.000	.000	.000	.000	.000	.000	.000
4	2	.080	.000	.000	.000	.000	.000	.000	.000
8	2	.025	.000	.000	.000	.000	.000	.000	.000
16	2	.007	.000	.000	.000	.000	.000	.000	.000
32	2	.002	.000	.000	.000	.000	.000	.000	.000
2	4	.420	.120	.017	.000	.000	.000	.000	.000
4	4	.132	.011	.001	.000	.000	.000	.000	.000
8	4	.024	.001	.000	.000	.000	.000	.000	.000
16	4	.003	.000	.000	.000	.000	.000	.000	.000
32	4	.000	.000	.000	.000	.000	.000	.000	.000
2	8	.627	.349	.162	.056	.013	.003	.001	.000
4	8	.243	.042	.005	.001	.000	.000	.000	.000
8	8	.033	.001	.000	.000	.000	.000	.000	.000
16	8	.001	.000	.000	.000	.000	.000	.000	.000
32	8	.000	.000	.000	.000	.000	.000	.000	.000
2	16	.785	.598	.437	.304	.197	.115	.058	.023
4	16	.429	.156	.045	.010	.002	.000	.000	.000
8	16	.076	.004	.000	.000	.000	.000	.000	.000
16	16	.002	.000	.000	.000	.000	.000	.000	.000
32	16	.000	.000	.000	.000	.000	.000	.000	.000
2	32	.884	.776	.675	.581	.494	.415	.343	.278
4	32	.631	.377	.211	.109	.050	.020	.007	.002
8	32	.200	.030	.003	.000	.000	.000	.000	.000
16	32	.007	.000	.000	.000	.000	.000	.000	.000
32	32	.000	.000	.000	.000	.000	.000	.000	.000
2	64	.940	.882	.825	.771	.718	.668	.619	.572
4	64	.786	.610	.465	.348	.255	.182	.127	.085
8	64	.407	.150	.049	.014	.003	.001	.000	.000
16	64	.043	.001	.000	.000	.000	.000	.000	.000
32	64	.000	.000	.000	.000	.000	.000	.000	.000
2	128	.969	.939	.909	.880	.852	.823	.796	.768
4	128	.885	.780	.684	.598	.520	.451	.388	.332
8	128	.622	.376	.220	.125	.068	.035	.017	.008
16	128	.169	.024	.003	.000	.000	.000	.000	.000
32	128	.002	.000	.000	.000	.000	.000	.000	.000
2	256	.985	.969	.954	.939	.924	.909	.894	.880
4	256	.940	.883	.828	.776	.727	.680	.635	.592
8	256	.784	.610	.471	.360	.273	.205	.153	.112
16	256	.389	.143	.050	.016	.005	.001	.000	.000
32	256	.029	.001	.000	.000	.000	.000	.000	.000

a new transaction is granted the locks requested when k transactions are already active:

$$q_k = \sum_{i=1}^{m-1} P_k(i) \sum_{p=1}^{m-i} X_i(p). \tag{6}$$

3.4 Discussions and Numerical Examples

The results obtained in the previous section may help us gain a first insight into the effect of locking on the operation of the transaction system. The mean number $\bar{l}_k$ of locked granules and the probability q_k give clear indications of the maximum possible number of simultaneously active transactions, and of the chance that a new transaction has its locks granted.

These indications will have to be analyzed and interpreted with respect to the model of transaction behavior that has been used. Recall that the model assumes that, for a given transaction, its accesses to the cells of the database are uniformly distributed over the whole set of cells: In other words, we assume that transaction references exhibit no locality. We may thus expect with this model of transactions behavior the effect of locking to be severe.

Those observations are illustrated by the numerical results presented in Tables I and II. These results have been obtained for $n = 2^i$, $i = 1, 5$, $m = 2^j$, $j = 1, 8$ and k ranging from 1 to 8. Table I and Table II, respectively, give $\bar{l}_k$ and q_k. Figures 3(a) and 3(b) depict the variation of $\bar{l}_k$ as a function of m with $n = 4$ and $n = 16$.

The importance of the no-locality assumption is revealed by the influence of the number n of accesses made by a transaction on the mean number of blocked granules $\bar{l}_k$. Transactions with a large number of accesses to the database very rapidly tend to lock most of the granules and to prevent activation of new transactions. On the other hand, this effect is partly alleviated by an increase in the number of granules. However, as we shall see in the following sections, the increase in the number of granules causes an increase of the CPU and I/O activity devoted to locks requests, which is detrimental to the processing rate of active transactions.

Table II. $\bar{l}_k$.

n	m	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
2	2	1.333	2.000	**********	**********	**********	**********	**********	**********
4	2	1.600	2.000	**********	**********	**********	**********	**********	**********
8	2	1.778	2.000	**********	**********	**********	**********	**********	**********
16	2	1.882	2.000	**********	**********	**********	**********	**********	**********
32	2	1.939	2.000	**********	**********	**********	**********	**********	**********
2	4	1.600	3.000	3.833	4.000	**********	**********	**********	**********
4	4	2.286	3.671	3.969	4.000	**********	**********	**********	**********
8	4	2.909	3.910	3.995	4.000	**********	**********	**********	**********
16	4	3.368	3.977	3.999	4.000	**********	**********	**********	**********
32	4	3.657	3.994	4.000	4.000	**********	**********	**********	**********
2	8	1.778	3.500	5.132	6.596	7.597	7.899	7.977	8.000
4	8	2.909	5.413	7.194	7.850	7.969	7.993	7.999	8.000
8	8	4.267	7.003	7.885	7.986	7.998	8.000	8.000	8.000
16	8	5.565	7.740	7.985	7.999	8.000	8.000	8.000	8.000
32	8	6.564	7.943	7.998	8.000	8.000	8.000	8.000	8.000
2	16	1.882	3.750	5.599	7.423	9.213	10.954	12.616	14.138
4	16	3.368	6.599	9.626	12.327	14.462	15.586	15.892	15.966
8	16	5.565	10.341	13.864	15.589	15.939	15.988	15.997	15.999
16	16	8.258	13.664	15.727	15.979	15.998	16.000	16.000	16.000
32	16	10.894	15.370	15.976	15.999	16.000	16.000	16.000	16.000
2	32	1.939	3.875	5.806	7.733	9.654	11.568	13.474	15.372
4	32	3.657	7.274	10.841	14.344	17.761	21.061	24.192	27.060
8	32	6.564	12.836	18.690	23.914	28.149	30.852	31.787	31.957
16	32	10.894	20.235	27.220	30.982	31.900	31.989	31.998	32.000
32	32	16.254	26.994	31.368	31.971	31.998	32.000	32.000	32.000
2	64	1.969	3.937	5.905	7.871	9.836	11.800	13.762	15.723
4	64	3.821	7.631	11.429	15.213	18.981	22.731	26.460	30.164
8	64	7.211	14.333	21.345	28.216	34.905	41.349	47.452	53.053
16	64	12.962	25.326	36.848	47.137	55.556	61.209	63.562	63.950
32	64	21.558	40.039	53.944	61.685	63.824	63.990	63.999	64.000
2	128	1.984	3.969	5.953	7.936	9.920	11.903	13.886	15.869
4	128	3.908	7.814	11.717	15.616	19.512	23.404	27.293	31.177
8	128	7.585	15.146	22.678	30.180	37.645	45.070	52.447	59.768
16	128	14.322	28.456	42.361	55.975	69.215	81.958	94.017	105.097
32	128	25.761	50.314	73.180	93.603	110.387	121.878	127.027	127.939
2	256	1.992	3.984	5.976	7.969	9.960	11.952	13.944	15.936
4	256	3.954	7.907	11.859	15.810	19.761	23.711	27.660	31.608
8	256	7.787	15.568	23.341	31.108	38.866	46.616	54.357	62.089
16	256	15.114	30.176	45.180	60.117	74.979	89.754	104.430	118.989
32	256	28.544	56.705	84.397	111.500	137.845	163.190	187.168	209.211

4. Organization and Operations of the CPU and I/O Resources

At a given instant, the transactions in states REQUEST or ACTIVE share the CPU and I/O resources of the system. We shall assume that there is no priority scheme set between ACTIVE and REQUEST transactions, and that the CPU processes these transactions on a processing-sharing (PS) fashion and I/O units process them according to a FIFO policy. The system consists of one central processing unit (CPU) and a set of B identical independent disk units (DU). The I/O operations of a transaction are uniformly distributed over the B DUs.

We shall now describe the different CPU and I/O operations performed by REQUEST and ACTIVE transactions.

4.1 Operations Performed by REQUEST Transaction

Let $v(m)$ be the mean number of distinct granules requested by a transaction. A transaction in the state REQUEST will thus perform on the average $v(m)$ elementary lock request operations. An elementary lock request operation consists of:

—a CPU service time with mean t_R;
—a DU service time with mean s_R.

An expression of $v(m)$ is simply obtained from the results presented in the previous section. We have

$$v(m) = \sum_{p=1}^{m} p \cdot x(p)$$

and, from eq. (1),

$$v(m) = \frac{mn}{m+n-1}. \tag{7}$$

It should be noted here that we have assumed that the elementary operations involved in a lock request, i.e., the CPU and DU services, do not depend on m. This is certainly an optimistic assumption since management of locks tables may become more costly as m increases, and, for instance, several disk accesses may be needed per lock request for a large value of m. Therefore, results

Fig. 3(a). $\bar{l}_k$ function of m, $n = 4$.

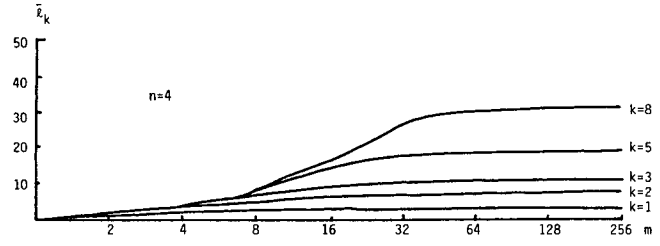


Fig. 3(b). $\bar{l}_k$ function of m, $n = 16$.

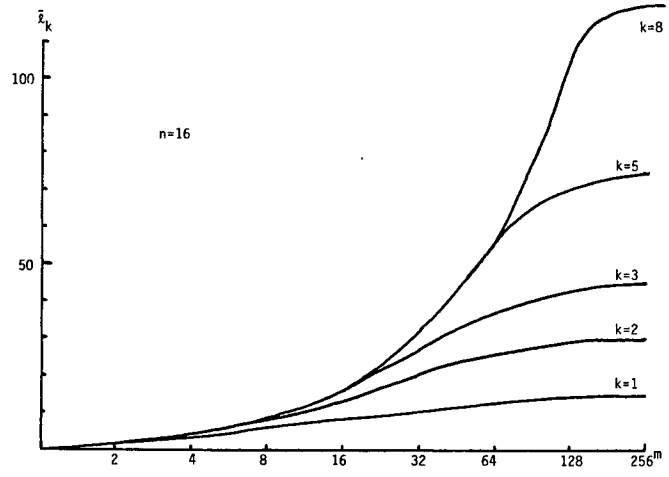
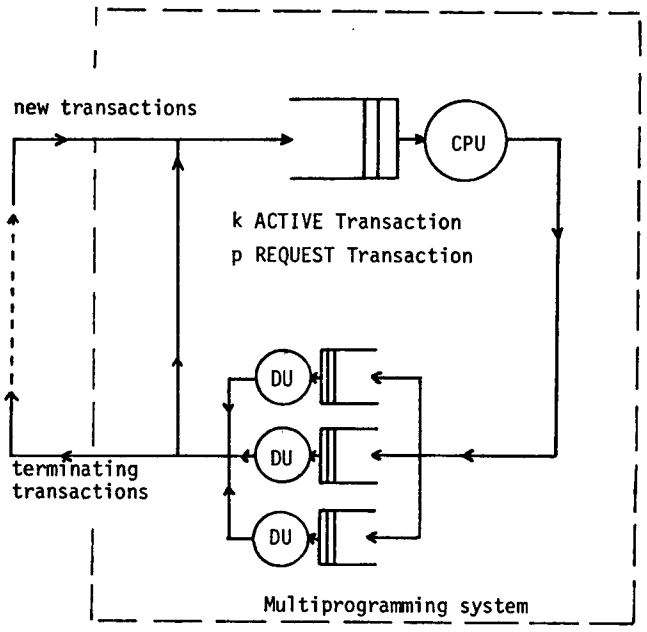


Fig. 4.



obtained for fixed values of t_R and s_R will have to be interpreted with respect to this assumption.

4.2 Operations Performed by Active Transactions

A transaction in the ACTIVE state performs n accesses to the entities of the database, each access being followed by a CPU processing time. The activity of an ACTIVE transaction will thus consist of n elementary operations comprising:

—a CPU service time with mean t_A;
—a DU service time with mean s_A.

We assume that the CPU and DU service times are independent random variables, and that DU service times have a negative exponential distribution.

4.3 Processing Rates of ACTIVE and REQUEST Transactions

Let k and p be, respectively, the number of ACTIVE and REQUEST transactions processed by the resources at a given instant of time. Due to the fact that ACTIVE and REQUEST transactions have different behaviors, their processing rates depend on k and p through functions of the form $\mu_A(k, p)$ and $\mu_R(k, p)$.

In order to derive $\mu_A(k, p)$ and $\mu_R(k, p)$, we follow the standard approach used for the throughput analysis of multiprogramming systems [4]. As represented in Figure 4, fixed numbers k and p of ACTIVE and REQUEST transactions are maintained in the system, and the sharing of the CPU and DU resources by the two types of transactions is analyzed using a closed queueing network model with different classes of customers [1].

An analysis of this model will yield for instance the throughput rate $X_A(k, p)$ and $X_R(k, p)$ of ACTIVE and REQUEST transactions through station CPU. Since ACTIVE and REQUEST transactions require on the average, respectively, n and $v(m)$ CPU service times, we have

$$\mu_A(k, p) = X_A(k, p)/n,$$
$$\mu_R(k, p) = X_R(k, p)/v(m).$$

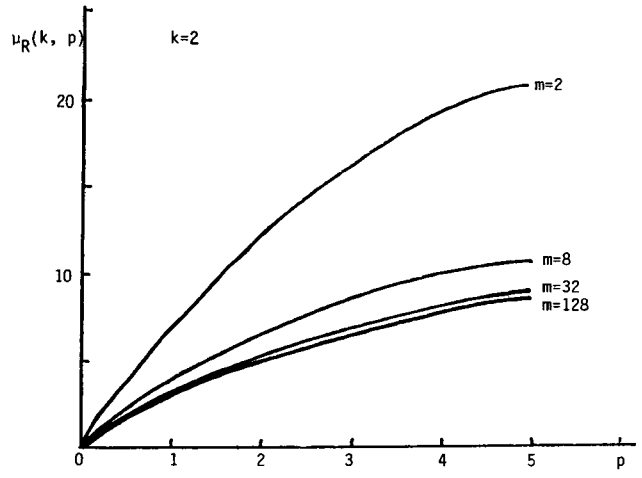
The functions $\mu_A(k, p)$ and $\mu_R(k, p)$ aggregate the different effects of resource sharing by ACTIVE and REQUEST transactions. Using the equivalent server approach [3], they will be used at modeling level 1 to analyze the global performance of the transaction system.

It should be noted that $\mu_A(k, p)$ and $\mu_R(k, p)$ depend on the granularity m through the function $v(m)$. Increasing the number of granules will increase the number of elementary lock requests performed by a transaction in the state REQUEST, and limit the availability of the CPU and DU resources to ACTIVE transaction. As a consequence, the processing rate of ACTIVE transactions will be degraded.

In order to illustrate this effect, we have plotted in Figure 5 $\mu_R(k, p)$ as a function of p for m ranging from 2 to 2^8 with $n = 4$ and $k = 2$. The results have been derived with the following values of the parameters:

$t_A = 50$ msec.
$t_R = 10$ msec.
$s_A = s_R = 50$ msec.
$B = 3$ (3 identical DUs).

Fig. 5. $\mu_R(k, p)$ function of p.



They have been obtained by using the solution package QNAP [6, 16] developed by CII-Honeywell Bull and INRIA. The listing of the QNAP program used for this computation and examples of its output are given in Table III. Note that the value of $1/\mu_A(k, p)$ and $1/\mu_R(k, p)$ are stored in an array after each resolution so as to be used later on when global performance is analyzed.

It can be observed from Figure 5 that, due to the specific behavior of the function $v(m)$, the effect of granularity on the locking overheads is particularly important as m increases from 2 to 2^5.

5. Analysis of Global Performance

Having analyzed modeling levels 2 and 3, we now dispose of the aggregate results that are needed for the analysis of the global performance. The interactions between the different factors which have already been investigated are summarized in Figure 6. Analysis will be conducted at modeling level 1 by using a multiclass queueing network model as presented in Figure 7. The set of K terminals is modeled by an infinite servers station TERMINAL with mean thinking time T, and the set of CPU-I/O resources is replaced by two exponential FIFO stations RESREQ and RESACT, with processing dependent rates $\mu_A(k, p)$, $(\mu_R(k, p))$ when k ACTIVE transactions and p REQUEST transactions are, respectively, present in these stations.

When a REQUEST transaction has completed its lock requests, its locks are granted with probability q_k, if k transactions are active, and it then enters the set of ACTIVE transactions; with probability $1 - q_k$, its locks are denied and the transaction proceeds to the BLOCKED queue.

Transactions are removed from the BLOCKED queue when ACTIVE transactions terminate and release their locks. Thus, the removal of transactions from the BLOCKED queue is triggered by departures from the station RESACT. Let r be the maximum number of transactions removed from queue BLOCKED. If i is the current number of transactions in queue BLOCKED, the actual number of transactions removed from queue BLOCKED at each departure is $\min(i, r)$.

Due to the strong dependence between queues BLOCKED, RESACT, and RESREQ, standard results on multiclass queueing network models are no longer applicable. Given the assumptions of the model, the only

Table III.

```
/DECLAR/QUEUE CPU,DU1,DU2,DU3;
      CLASS ACTIVE,REQUEST;
      REAL XA,XR,Y,NA,NR,TA(100),TR(100),M,N;
       INTEGER KTR,IA,IR,I;
&
/STATION/NAME=CPU;
      SCHED=PS;
      SERVICE(ACTIVE)=EXP(XA);
      SERVICE(REQUEST)=EXP(XR);
      TRANS=DU1,1,DU2,1,DU3,1;
      INIT(ACTIVE)=IA;
      INIT(REQUEST)=IR;
&
/STATION/NAME=DU1;
       SERVICE=EXP(Y);
      TRANS=CPU;
&
/STATION/NAME=DU2;COPY=DU1;
&
/STATION/NAME=DU3;COPY=DU1;
&
&
/PARAM/CLASS=CPU;
/EXEC/BEGIN
      XA:=0.05;
       XR:=0.01;
      N:=4;
      M:=1;
&
      NA:=4;
      NR:=M*N/(M+N-1);
&
      Y:=0.05;
      KTR:=5;
&
      IA:=0;
      WHILE IA<=KTR DO BEGIN
            IR:=0;
             WHILE IR<=KTR DO BEGIN
             ANALYTIC;
             I:=(KTR+1)*IA+IR+1;
             TA(I):=NA/MTHRUPUT(CPU,ACTIVE);
             TR(I):=NR/MTHRUPUT(CPU,REQUEST);
             PRINT(I,IA,IR,TA(I),TR(I));
            IR:=IR+1;
                  END;
          IA:=IA+1;
          END;
      END;
```

```
        **** ANALYTIC RESOLUTION ****
****SUBCHAIN   1 TOTAL NUMBER OF CUSTOMERS=   3
****SUBCHAIN   2 TOTAL NUMBER OF CUSTOMERS=   5
*****************************************************************
* NAME   * SERVICE * BUSY PCT *  CUST NB * RESPONSE *  THRUPUT *
*****************************************************************
*        *         *          *          *           *          *
*  CPU   * .2027E-01* .8535    * 2.637    * .6262E-01* 42.12    *
* (ACTI) * .5000E-01* .5404    * 1.528    * .1413    * 10.81    *
* (REQU) * .1000E-01* .3131    * 1.110    * .3545E-01* 31.31    *
*        *         *          *          *           *          *
*  DU1   * .5000E-01* .7020    * 1.788    * .1273    * 14.04    *
*        *         *          *          *           *          *
*  DU2   * .5000E-01* .7020    * 1.788    * .1273    * 14.04    *
*        *         *          *          *           *          *
*  DU3   * .5000E-01* .7020    * 1.788    * .1273    * 14.04    *
*        *         *          *          *           *          *
*****************************************************************
SPACE USED :   3/100
        **** END OF ANALYTIC RESOLUTION ****
```

Fig. 6.

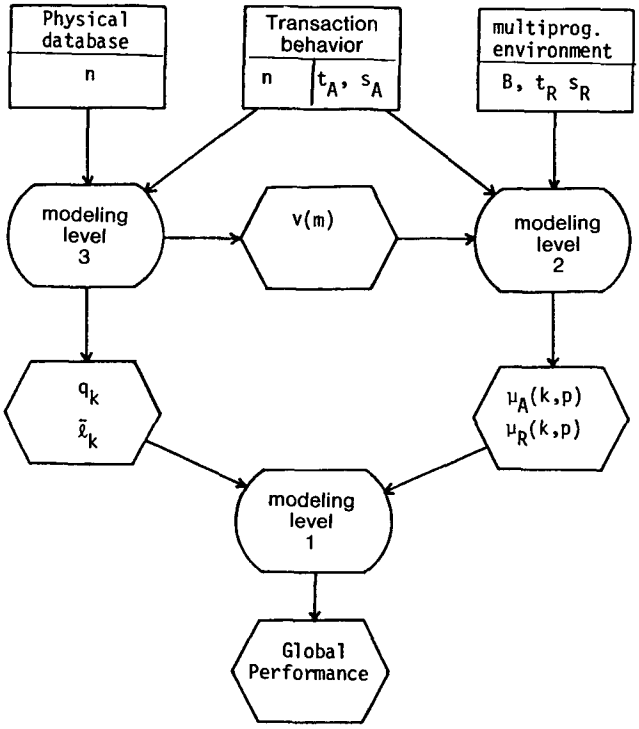
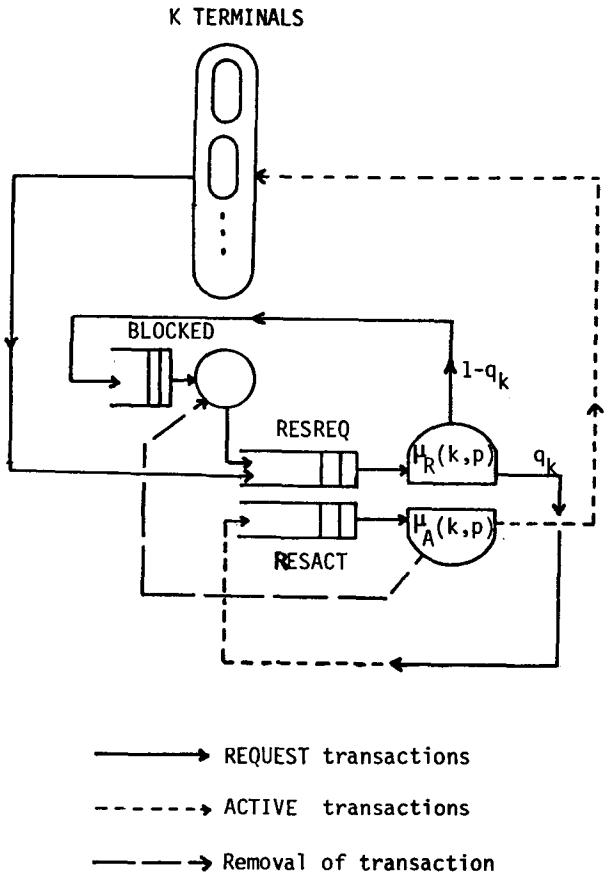


Fig. 7.



exact solution technique available is through Markovian analysis [14]. This is done by again using the solution package QNAP. The text of the QNAP program is given in Table IV. Transfers from the BLOCKED queue to the RESREQ queue are specified by the primitive MOVE. Since QNAP uses only linear arrays, a macroinstruction IND(I) is used to access the element of the arrays TA and TR where $1/\mu_A(k,p)$ and $1/\mu_R(k,p)$ are stored.

The results are presented in Figures 8 and 9. They have been obtained with the following values of the parameters of the model:

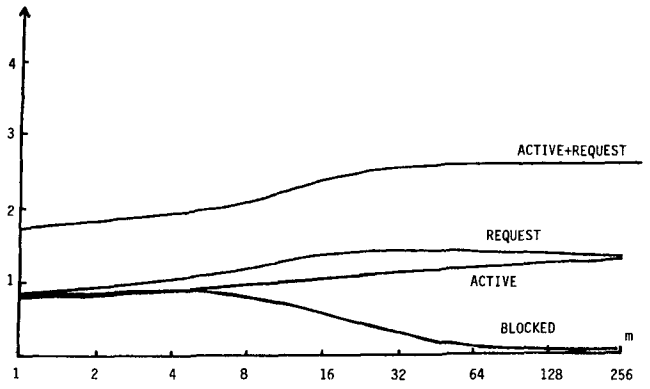
$n = 4$,
$m = 2^0$ to 2^8,
$t_A = 50$ msec.,
$t_R = 10$ msec.,
$s_A = s_R = 50$ msec.,
$B = 3$,
$r = 1$,
$T = 1.5$ sec.

Figure 8 displays the mean number of ACTIVE, REQUEST, and BLOCKED transactions in the system as a function of m. The main observation is that as m increases, the mean number of BLOCKED transactions decreases rapidly, while the mean number of ACTIVE transactions steadily increases.

Table IV.

```
/DECLAR/QUEUE TERMINAL,RESREQ,RESACT,BLOCK;INTEGER
        K1,K2;REAL TREF,TREQUEST,TACTIVE,QQ;
        REAL Q(10)=1.0,0.94,0.883,0.828,0.776,0.727,0.68,0.635
                        ,0.592,0.332;
&
$ MACRO IND(I)
        IA:=CUSTNB(RESACT);
        IR:=CUSTNB(RESREQ);
        I:=(KTR+1)*IA+IR+1;
$ END
&
/STATION/NAME=TERMINAL;
         TYPE=DELAY;
         SERVICE=EXP(TREF);
         TRANS=RESREQ(REQUEST);
         INIT(ACTIVE)=KTR;
/STATION/NAME=RESREQ;
      SERVICE=BEGIN
                 I:=CUSTNB(RESACT);
                 QQ:=Q(I+1);
                 $IND(I)
                 EXP(TR(I));
               END;
      TRANSIT=RESACT(ACTIVE),QQ,BLOCK;
/STATION/NAME=RESACT;
      SERVICE=BEGIN
                 K1:=1;
                 $IND(I)
                 EXP(TA(I));
               END;
      TRANSIT=TERMINAL;
/PARAM/OPTION=RESULT;TEST=BEGIN
      IF K1=1 THEN  BEGIN K1:=0;MOVE(BLOCK,RESREQ);END;
       END;
/EXEC/BEGIN
       NETWORK(TERMINAL,RESREQ,RESACT,BLOCK);
      K1:=0;
      TREF:=1.5;
      MARKOV;
      PRINT(NCUSTNB(BLOCK),NTHRUPUT(BLOCK),NRESPONSE(BLOCK));
      END;
```

Fig. 8. Mean number of transactions in queues.



The mean number of REQUEST transactions reached a maximum for $m = 32$ and slowly decreases afterward. It is thus to be expected that the global throughput will increase with m for large values of m.

This is illustrated in Figure 9 where the throughput $D(m)$ of the system is plotted, for n ranging from 2 to 2^5, expressed in terms of the mean number of transactions processed per unit of time. It is observed that $D(m)$ decreases through a minimum and then increases with m.

This observation, which contradicts the results presented in [9] where $D(m)$ exhibited a maximum, is to be interpreted with respect to the set of assumptions that have been used throughout the analysis. As it appears from the results of modeling levels 2 and 3, since q_k increases with m, the ratio of the number of ACTIVE transactions to the number of REQUEST transactions tends to increase; on the other hand, since the mean number of locks $v(m)$ tested by REQUEST transactions increases with m, REQUEST transactions tend to need more processing as m increases. However, as indicated by the behavior of the throughput function $D(m)$, this behavior is determined by the increase of q_k rather than by the increase of $v(m)$. This points out again the key role played by locking overheads, here quantified by the function $v(m)$. In order to explain the discrepancies observed between our results and those presented in [9], it should be recalled that the function $v(m)$ used in this study is of the form $v(m) = \alpha m$. In general, assumptions on transaction behavior used in those two studies are basically different: Ries and Stonebraker use a "linear" model of transaction behavior; we use a "nonlinear" model.

6. Conclusions

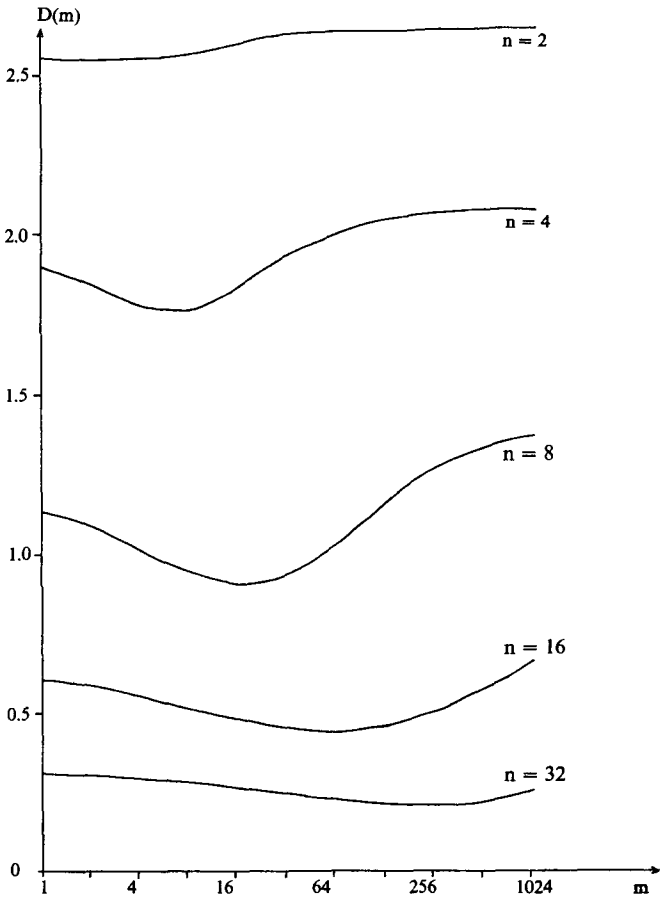
We have presented in this paper an analytical framework for the performance analysis of locking mechanisms in transaction systems, and we have illustrated this approach by a detailed analysis based on a simple probabilistic model of transaction behavior.

This analysis provides a clear understanding of the various factors that determine global performance. It also raises many new issues that can only be solved by further extensive experimental and analytical studies. Two particular topics deserve special attention: the modeling of transaction behavior and the modeling of locking overheads. As noted above, the model of transaction behavior we have used makes a strong assumption on the distribution of references over the database: This model has to be validated by comparing its results to those of more complex models, involving, for instance, localities or sequentialities in the reference pattern. We have also pointed out in Section 5 the key role played by locking overheads. Here again a more refined analysis is needed, in relation to the modeling of transaction behavior. We are currently working along those directions and we hope that this paper will initiate parallel investigations.

Acknowledgments. The authors would like to thank J.L. Colomer of the Facultad de Informatica, Universidad de Barcelona, Spain; R. Balter of the Centre Scientifique CII-Honeywell Bull in Grenoble, France; and M. Scholl of INRIA for their helpful contributions.

Received 2/80; accepted 4/80; revised 6/80

Fig. 9.



References

1. Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, J. Open, closed and mixed networks with different classes of customers. *J. ACM 22*, 2 (April 1975), 248–260.
2. Blasgen, M., Gray, J., Mitoma, M., and Price, T. The convoy phenomena. *Operating Syst. Rev. 3*, 2 (April 1979), 20–25.
3. Chandy, K.M., and Sauer, C.H. Approximate methods for analyzing queueing network models of computing systems. *Comptng. Surveys 10*, 3 (Sept. 1978), 281–317.
4. Denning, P.J., Kahn, K.C., Leroudier, J., Potier, D., and Suri, R. Optimal multiprogramming. *Acta Inform. 7*, 2 (1976), 197–216.
5. Gilbert, D.C. Modelling spin locks with queueing networks. *Operating Syst. Rev. 2*, 1 (Jan. 78), 29–42.
6. Merle, D., Potier, D., and Veran, M. A tool for computer system performance analysis. Proc. Internat. Conf. on Performance of Comptr. Installations 1978, North-Holland Pub. Co., Amsterdam, 1978, pp. 195–214.
7. Muntz, R.R., and Krenz, G.K. Concurrency in database systems—simulation study. Proc. ACM SIGMOD Internat. Conf. on Management of Data, Toronto, Ontario, Canada, August 1977.
8. Riordan, J. *An Introduction to Combinatorial Analysis.* John Wiley and Sons, N.Y., 1958.
9. Ries, D.R., and Stonebraker, M.R. Effects of locking granularity in a database management system. *Trans. Database Systs. 2*, 3 (Sept. 1977), 233–246.
10. Ries, D.R., and Stonebraker, M.R., Locking granule revisited. *Trans. Database Systs. 4*, 2 (June 1979), 210–227.
11. Rodriguez-Rosell, J. Empirical data reference behaviour in database systems. *Computer 9*, 11 (Nov. 1976), 9–13.
12. Rodriguez-Rosell, J., and Hildebrand, D. A framework for evaluation of database management system. Proc. Internat. Comptng. Symp. 1975, E. Gelenbe and D. Potier, Eds., North-Holland Pub. Co., Amsterdam, 1975.
13. Smith, A.J. Sequentiality and prefetching in database systems. *Trans. Database Systs. 3*, 3 (Sept. 1978), 223–247.
14. Stewart, W.J. A comparison of numerical techniques in Markov modelling. *Comm. ACM 21*, 2 (Feb. 1978), 144–151.
15. Tuel, W.G., and Rodriguez-Rosell, J. A methodology for evaluation of database systems. Res. Rep. RJ 1688, IBM Thomas J. Watson Res. Ctr., Yorktown Heights, N.Y., 1976.
16. Veran, M. QNAP description language. Res. Rep., Antenne Scientifique CII-Honeywell-Bull, Grenoble, France, 1979.

Professional Activities Calendar of Events

ACM's calendar policy is to list open computer science meetings that are held on a not-for-profit basis. Not included in the calendar are educational seminars, institutes, and courses. Submittals should be substantiated with name of the sponsoring organization, fee schedule, and chairman's name and full address.

One telephone number contact for those interested in attending a meeting will be given when a number is specified for this purpose.

All requests for ACM sponsorship or cooperation should be addressed to Louis Fiora, Conference Coordinator, ACM Headquarters, 1133 Avenue of the Americas, New York, NY 10036; 212 265-6300. For European events, a copy of the request should also be sent to the European Representative. Technical Meeting Request Forms for this purpose can be obtained from ACM Headquarters. Lead time should include 2 months (3 months if for Europe) for processing of the request, plus the necessary months (minimum 3) for any publicity to appear in *Communications.*

■ This symbol indicates that the Conferences and Symposia Committee has given its approval for ACM sponsorship or cooperation. Chairman of the Conferences and Symposia Committee is Seymour J. Wolfson, 643 MacKenzie Hall, Wayne State University, Detroit, MI 48202.

New Listings are shown first; they will appear next month as Previous Listings.

NEW LISTINGS

6–8 November 1980

SIAM Fall 1980 Meeting, Houston, Tex. Sponsor: Society for Industrial and Applied Mathematics. Contact: H.B. Hair, 33 South 17th Street, Philadelphia, PA 19103; 215 564-2929.

26–28 February 1981

8th Annual Conference of Mid-South Association for Educational Data Systems, Memphis, Tenn. Sponsor: Mid-South AEDS. Prog. chm: Lloyd D. Brooks, Office Administration, Memphis State University, Memphis, TN 38152; 901 454-2453.

25–27 March 1981

Conference on Information Sciences and Systems, Baltimore, Md. Sponsor: Johns Hopkins University. Prog. Dirs: Gerard G. L. Meyer and Wilson J. Rugh, EE Dept., Johns Hopkins University, Baltimore, MD 21218; 301 338-7003.

19–25 April 1981

Formalization of Programming Concepts, Peniscola, Spain. Sponsor: European Association for Theoretical Computer Science. Contact: ICFPC, Facultat d'Informàtica, Universitat Politècnica de Barcelona, Jordi Girona Salgado 31, Barcelona 34, Spain.

11–13 May 1981

■ **Thirteenth Annual ACM Symposium on Theory of Computing,** Milwaukee, Wis. Sponsors: ACM SIGACT, University of Wisconsin, Milwaukee. Contact: George Davida. Dept. of EECS, University of Wisconsin, Milwaukee, WI 53201.

11–15 May 1981

International Symposium COMNET 81: Networks from the Users' Point of View, Budapest, Hungary. Sponsors: IFIP, UNESCO. Contact: COMNET 81 Secretariat, John V. Neumann, Society for Computer Sciences, P.O.B. 240, H-1368 Budapest, Hungary.

17–20 May 1981

■ **Fifth International Conference on Computers and the Humanities.** Ann Arbor, Mich. Sponsor: Assoc. for Computers and the Humanities, Assoc. for Literary and Linguistic Computing, University of Michigan in cooperation with ACM SIGLASH. Contact: Richard W. Bailey, Dept. of English, University of Michigan, Ann Arbor, MI 48104.

18–22 May 1981

Workshop on Recursion Theoretic Aspects of Computer Science, W. Lafayette, Ind. Sponsor: National Science Foundation. Contact: Carl Smith, Dept. of Computer Sciences, Purdue University, W. Lafayette, IN 47907; 317 749-6301.

20–22 May 1981

■ **Third Conference on Databases in the Humanities and Social Sciences.** Ann Arbor, Mich. Sponsor: Assoc. for Computers and the Humanities, University of Michigan. Contact: Gregory A. Marks, Institute for Social Research, University of Michigan, Ann Arbor, MI 48104.

31 May–2 June 1981

■ **Fourth Annual International SIGIR Conference,** Berkeley, Calif. Sponsor: ACM SIGIR in cooperation with ASIS. Conf. chm: William S. Cooper, School of Library and Information Studies, University of California, Berkeley, CA 94720; 415 642-4697 or 1464.

29 June–1 July 1981

■ **18th Design Automation Conference,** Nashville, Tenn. Sponsor: ACM SIGDA, IEEE-CS. Conf. chm: Robert J. Smith II, V-R Information Systems Inc., 5758 Balcones Drive, Suite 205, Austin, TX 78731; 512 458-8131.

29–31 July 1981

■ **The Computing Environment for Mathematical Software,** Pasadena, Calif. Sponsors: JPL, ACM SIGNUM. Conf. chm: Fred T. Krogh, 125/128, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91103; 213 354-6127.

31 August–4 September 1981

10th International Symposium on Mathematical Foundations of Computer Science, Strbské Pleso, Czechoslovakia. Sponsor: Computing Research Centre, Bratislava. Symp. chm: Jozef Gruska, Computer Research Centre, Dúbravská 3, 885 31 Bratislava, Czechoslovakia.

7–9 September 1981

■ **International Conference on Computer Hardware Description Languages and Their Applications,** Kaiserslautern, Fed. Rep. of Germany. Sponsors: IFIP Tech. Comm. TC 10 and its Working Group WG 10.2 in cooperation with ACM SIGARCH and SIGDA, IEEE-CS, GI, NTG. Conf. chm: Reiner Hartenstein, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, D-6750 Kaisenslautern, F.R. Germany.

PREVIOUS LISTINGS

15–17 October 1980

V ICCRE, Fifth International Conference on Computers in Chemical Research and Education, Toyohashi, Japan (a post congress symposium of Seventh International CODATA Conference, Kyoto, Oct. 8–11). Contact: S. Sasaki, School of Materials Science, Toyohashi University of Technology, Tempaku, Toyohashi, Japan 440.

15–18 October 1980

Symposium on Optimization Methods—Applied Aspects, Varna, Bulgaria. Sponsors: IFAC, IFORS. Contact: National Centre for Cybernetics & Computer Techniques, Committee for Science, 8 Slavyanska St., Sofia, Bulgaria.

17 October 1980

Conference on Computer Graphics and Microcomputers, Cambridge, Mass. Sponsor: New England Regional Computing Program. Contact: Rita Donahue, NERComP, 385 Elliot St., Newton Upper Falls, MA 02164.

20 October 1980

National Information Systems: Getting Ready for 1984, Washington, D.C. Sponsor: Computer Law Association. Contact: Michael Yourshaw, 1776 K St., N.W., Washington, DC 20006; 202 857-5029.

20–22 October 1980

CPEUG 80, 16th Meeting of the Computer Performance Evaluation Users Group, Orlando, Fla. Sponsor: NBS. Contact: Theodore F. Gonter, U.S. General Accounting Office, 441 G. St., N.W., Room 6118, Washington, DC 20548; 202 275-5410.

20–22 October 1980

Texas Association for Educational Data Systems 1980 Annual Convention, Austin, Tex. Sponsor: TAEDS. Contact: Phil Gensler, Dept. of CIS, West Texas State University, Canyon, TX 79016.

20–24 October 1980

International Seminar on Software Engineering Applications, Capri (Napoli), Italy. Sponsors: ACM Italian Chapter, AICA, ENIDATA and others. Contact: Marilena Vendramin, Direzione Tecnica, Systems and Management, Via Medici, 2, 20123 Milan, Italy.

21–23 October 1980

1980 Biennial Display Research Conference, Cherry Hill, N.J. Sponsors: IEEE Electron Devices Society, SID, Advisory Group on Electronic Devices. Contact: Thomas Henion, Palisades Institute, 201 Varick Street, New York, NY 10014.

21–24 October 1980

SEARCC 80, Jakarta, Indonesia. Sponsor: South East Asia Regional Computer Confederation. Contact: SEARCC 80 Conference Implementation Committee, P.O. Box 4487, Jakarta, Indonesia.

26–29 October 1980

DPMA International Conference, Philadelphia, Pa. Sponsor: Data Processing Management Association. Contact: Bill Zalud, DPMA, 505 Busse Highway, Park Ridge, IL 60068.

27–29 October 1980

■ **ACM 80,** Nashville, Tenn. Sponsor: ACM. Gen. chm: Charles Bradshaw, Computer Center, Box

(*Calendar continued on p. 596*)