Mean response times for optimistic concurrency control in a multi-processor database with exponential execution times

Eindhoven University
of Technology

Department of Mathematics and Computing Science

Memorandum COSOR 95-43

Mean Response Times for Optimistic Concurrency Control in a Multi-Processor Database with Exponential Execution Times

S.A.E. Sassen
J. van der Wal
M.P. Bodlaender

Eindhoven, December 1995
The Netherlands

Mean Response Times for Optimistic Concurrency Control in a Multi-Processor Database with Exponential Execution Times [1]

S.A.E. Sassen, J. van der Wal, and M.P. Bodlaender

Department of Mathematics and Computing Science
Eindhoven University of Technology

December 22, 1995

Abstract

Using a non-productform queueing network model, two approximations are developed for computing the average response time of transactions in a multi-processor shared-memory database system with optimistic concurrency control. The time and resources needed for the validation of transactions are explicitly taken into account in the queueing model, since they are not always negligible. The performance of the approximations, tested against a simulation of the queueing model, is very good.

1 Introduction

In this paper we use queueing models to evaluate the performance of optimistic concurrency control in a multi-processor shared-memory real-time database. Concurrency control is the control of simultaneous execution of transactions such that overall correctness of the database is maintained, see e.g. Papadimitriou [1986]. The two basic concurrency control schemes are locking and optimistic concurrency control.

Under the locking scheme, an executing transaction holds a lock on all data-items it needs for the execution, thus introducing lock waits for transactions that conflict (i.e. have one or more data-items in common) with it. Consistency is guaranteed, however locking can cause long transaction waiting and response times and a low utilisation of the available hardware resources (the CPUs).

When the probability of transactions having a conflict is low and resources are not scarce, it can be advantageous[2] to use the optimistic concurrency control (OCC) scheme, that works without locks. Under optimistic concurrency control (Kung and Robinson [1981]), transactions processed by a database go through three phases: an execution phase, a validation phase and a commit phase. In the execution phase a transaction, say T, accesses any data-item it needs for the execution, regardless of the number of transactions already using that data-item. After the execution phase the validation phase is started, in which all data-items used by T are checked on read/write (R/W) or write/write (W/W) conflicts. A R/W conflict occurs when T has read a data-item which in the meantime has been overwritten by another transaction, or which is to be overwritten by a transaction that has already started its commit phase. A W/W conflict occurs when T wants to overwrite the same data-item as another transaction that is still in its commit phase. If either one of these conflicts occurs, T must be rerun. If no conflicts occur, T need not be rerun and enters the commit phase, where the

[1] This research is supported by the Technology Foundation (STW), project EIF33.3129

[2] this depends heavily on the characteristics of the transactions and the specific hardware environment, see Agrawal *et al.* [1987]

data-items used by T are updated.

The real-time database considered in this paper is managed by OCC. Since there are no locks involved in OCC, transactions never have to wait for data-items still in use by other transactions: data-access can take place immediately. The only waiting that occurs under OCC is waiting by a transaction for a CPU to become available for its execution. The absence of locks on data-items enables us to use a relatively simple queueing network to describe transaction processing under OCC. From the queueing network, approximations can be obtained for the average response time of a transaction, and for the average probability that a transaction has to be rerun.

In the past, others have also obtained analytic approximations of the average response time of transactions in a multi-processor shared-memory database with OCC, for example Menascé and Nakanishi [1982], Morris and Wong [1985], Kleinrock and Mehović [1992], and Yu *et al.* [1993]. They have all ignored the effect of concurrency control overhead on the response time, some by claiming it is negligible. However, in real-time databases with very small execution times, the time needed for validation may not be negligible compared to the execution time. Important examples of such systems are telecommunications systems. In this paper we include concurrency control overhead in the model. Omitting the time and resources needed for concurrency control leads to a simple, single-queue multi-server model, as opposed to the queueing network model that we obtain when accounting for concurrency control overhead. The observation that under OCC several transactions can execute at the same time but (for consistency reasons) only one transaction can validate at any time, leads to a queueing network with a multi-server station for the execution phase and a single-server station for the validation phase of the transactions. If the validation time is not very small compared to the execution time, the queue at the single-server validation station can grow very large which has a dramatic effect on the total response time of a transaction.

The outline of this paper is as follows. We first give a specification of the hardware in section 2, after which we translate this specification into a queueing network model in section 3. Section 4 contains two approaches for an approximative stochastic analysis of the queueing model. The first approach uses the steady-state probabilities of the (non-productform) queueing network to find an expression for the average response time of a transaction. However if the number of processors in the database system increases, the large number of states in the queueing model makes the computation of the state probabilities from a system of linear equations practically infeasible. The second approach is less elaborate since it avoids the numerical solution of a large system of equations. An intuitively appealing assumption about the commit process of transactions leads to a recursive procedure for approximating the average probability that a transaction has to be rerun. This recursive procedure uses the steady-state probabilities of a product-form queueing network, so it is easily implemented regardless of the number of processors in the database system. In section 5 we give numerical results for both approaches and compare them to a simulation of the queueing system. Finally, in section 6 we present the conclusions.

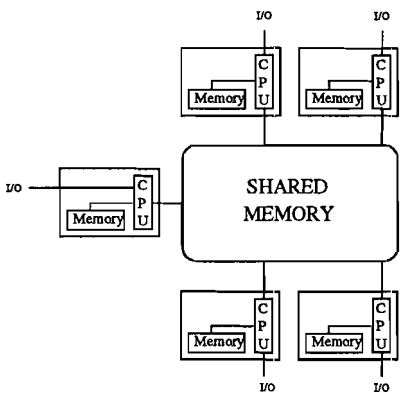


Figure 1: HARDWARE CONFIGURATION FOR $N = 4$

2 Hardware Specification

The hardware configuration of the database system that will be analysed in this paper is shown in Figure 1. It consists of $N + 1$ independently operating, tightly-coupled CPUs that can communicate with a shared memory. We assume that the communication network is such that the shared memory can be accessed by any number of CPUs simultaneously, and that the access time is not influenced by how many CPUs access the shared memory at the same time (this is called uniform memory-access). Each CPU has a private working memory and I/O capabilities. We assume that the shared memory is large enough to store all data and waiting transactions, so no external storage device (e.g. hard-disk) is attached to the system. Configurations with $N = 8$ are currently feasible.

3 From Specification to a Queueing Model

The multi-processor, shared-memory database with OCC as described above results in a queueing model of which a graphical representation is given in Figure 2.
Let us briefly explain how this queueing representation is obtained. We assume that transactions arrive at the database system according to a Poisson process with rate λ, i.e., the time between two subsequent arrivals is exponentially distributed with mean $1/\lambda$. An arriving transaction enters the database system at a dedicated CPU. We assume that there is only one such CPU, but this can be extended to several CPUs if necessary to cope with the stream of arriving transactions. The dedicated CPU has to

1. find a place in the shared memory for the transaction, and assign a sequence number
2. copy the transaction into the shared memory.

The remaining N CPUs (denoted by $\mathrm{CP}_1 \ldots \mathrm{CP}_N$) are used for the actual execution of the transactions.

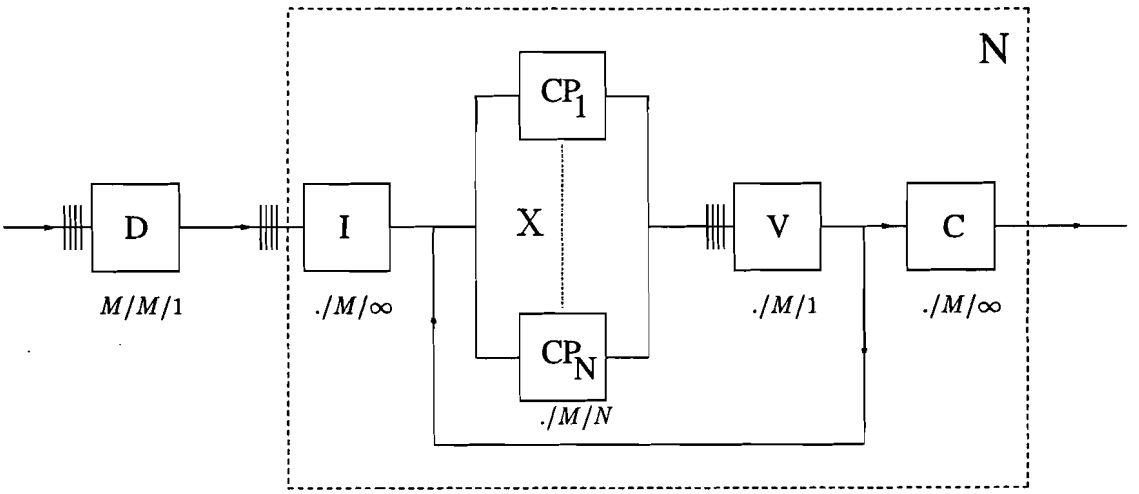


Figure 2: A QUEUEING MODEL FOR THE REAL-TIME DATABASE

A transaction's path through the system

D The dedicated CPU can only handle one transaction at a time, so transactions finding the CPU busy have to wait in a queue. In the queueing model this is represented by a single server with a first-come first-served (FCFS) queueing discipline. For ease of analysis we will assume that the service time at the dedicated CPU is exponentially distributed (as we will do with all service times). The mean service time needed by the dedicated CPU to handle a transaction is $1/\mu_0$.

Queue outside box Once a transaction is available in the shared memory, it must wait until one of the CPUs is free and retrieves it for execution. Transactions are retrieved in a FCFS fashion, according to their sequence numbers. Transactions waiting for their execution are represented in Figure 2 as a queue in front of the dashed box.

I We assume that the transactions cannot be split into subtransactions, so the entire execution of a transaction takes place at one of the CPUs. Before the execution of a transaction can start at a CPU, the transaction has to be copied to the CPU's private memory. This initialization can be done by several CPUs (thus for up to N transactions) at the same time so is represented in the queueing model by infinite service station I.

X Once the transaction is in the private memory of a CPU, the execution of the transaction starts. Data-items needed by the transaction are copied from shared memory to the CPU during the execution. Because N CPUs can handle transactions concurrently, the execution phase is modelled as a multi-server $./M/N$ queue. Since there cannot be more than N transactions inside the box, there never is a waiting queue at the multi server. Therefore we refer to service station X as an ample server. The execution time of a transaction at this server is taken exponentially distributed with mean $1/\mu_1$.

V When a transaction has finished execution, it goes through a validation phase. Only one transaction can validate at a time because the validation contains a small critical

section to prevent inconsistency. Thus, although the transaction is actually validated at its own CPU, we represent the validation phase by a single server with a FCFS queueing discipline. Note that there can be at most $N-1$ transactions waiting at that single server. We take the service time (validation time) exponentially distributed with mean $1/\mu_2$.

C If a transaction validates correctly, it enters the commit phase. The transaction writes the new values of the data-items into the shared memory and stores some validation information in shared memory. All this can be done by several CPUs at the same time so is represented by an infinite server. When finished, the transaction 'leaves' the database system and the CPU is freed. If necessary, the I/O channel of the CPU where the execution took place can be used to generate output for the users of the database. Note that the validation ensures that no two transactions that want to overwrite the same data-item can be at the commit server simultaneously.

Feedback arrow When the validation reveals that the transaction has to be rerun (because it conflicts with a committing or recently committed transaction), the transaction executes and validates again (at the same CPU). This goes on until finally a validation permits the transaction to commit.

All service times in the model are assumed to be exponentially distributed. However, it is not necessary to make this assumption for the service times at the ample servers X, I, and C. The second approximative approach that we describe below can easily be adapted for the situation with generally distributed execution times at the ample servers. The first approach however does require the assumption of exponential service times.

4 Performance Analysis

The objective of this study is to find a good approximation for the average performance of the database as a function of the system parameters (number of CPUs, execution time of a transaction, arrival rate of transactions) and to study the effect of concurrency control overhead on the database performance. The performance of the database system is measured in terms of the mean response time of an arbitrary transaction, and the long-run average probability that a transaction has to be rerun.

The queueing representation of the database system does not allow for an exact analysis of the response time. This is because we would then have to include into the state description exactly which data-items are needed by which transactions and also for each transaction information about if the data-items read have been changed by a recently committed transaction. However, this enormous state description is practically infeasible for doing computations.

Instead, we will perform an approximative analysis of the queueing model, using a much smaller state space. A probabilistic model is used for the occurrence of conflicts. Let p be the probability that two transactions conflict. We colour all transactions green on entering the dashed box. During its execution or during the time spent in the queue waiting for validation, a transaction is marked red with probability p whenever another transaction starts its commit phase. A red transaction always discovers at its validation that it cannot commit and must be rerun, a transaction that is still green at the start of its validation has had no conflicts

so is allowed to commit. The state description needed in the queueing model then consists of the number of transactions present at the dedicated CPU, the number of red and green transactions present at every infinite or ample service station, and the sequence of red and green transactions present at the single-server validation station.

In addition to the modelling assumption of probabilistic conflicts, we make the so-called **fake-restart** assumption, see Agrawal *et al.* [1987]. This is the assumption that at every rerun the transaction is replaced by a new, independent transaction whose execution and validation times are independent of the times of the previous run. This assumption is reasonable if the number of restarts per unit time is not too large and the load of the system is not too high.

Comparison of (a simulation of) the real database system with a simulation of this simpler system with red and green transactions and probabilistic conflicts should provide a conclusion about whether the simpler representation is accurate enough.

In the remainder of this text, 'the queueing network of Figure 2' always refers to the queueing model with probabilistic conflicts, unless mentioned otherwise.

The queueing network of Figure 2 with as state description the number of (green) transactions at the dedicated CPU, the number of red and green transactions at the infinite and ample service stations, and the sequence of red and green transactions present at the single-server station, is not a product-form queueing network. It is a network with two classes of customers (red, green transactions) and exponential service times, even with the same mean service time for both classes at the single FCFS station. Moreover there is Markovian routing for each customer class. However the BCMP conditions (Baskett *et al.* [1975]) are violated by the fact that one service completion at the validation server can change the class of a number of customers at the same time, and also by the blocking that occurs in front of the dashed box.

The approach we adopt for analysing the queueing model is a **decomposition–aggregation approach**. First we approximate the mean response time of a transaction in the dashed box, treating it as a closed queueing network with a constant population of k customers. (The population is kept constant at k by admitting a new transaction to the box as soon as another transaction has committed.) Doing this for all $k \leq N$ yields the mean response time in the box given a population of k customers, as will be seen in section 4.1. Next we consider the box as a single (exponential) service station with a service rate dependent on the number of transactions present in front of and inside the box, such that we can approximate the mean response time of a transaction on its complete path through the system. The state-dependent service rate of the box follows from the analysis of the closed system with k transactions. This aggregation step is performed in section 4.2.

4.1 The Closed System

Approximating the time a transaction spends in the dashed box (in the sequel, we refer to this box as the closed system) of Figure 2 is of most concern. First we simplify the analysis by removing station I from the box and changing the service time of station C to that of I plus C. This will not affect our approximation for the mean response time in the box, but does ease the analysis since the box now contains one less service station. So the original closed system is equivalent to the one having as first station ample server X, as second station single server V and as third station infinite server IC (with service rate μ_3). This system is illustrated by Figure 3.

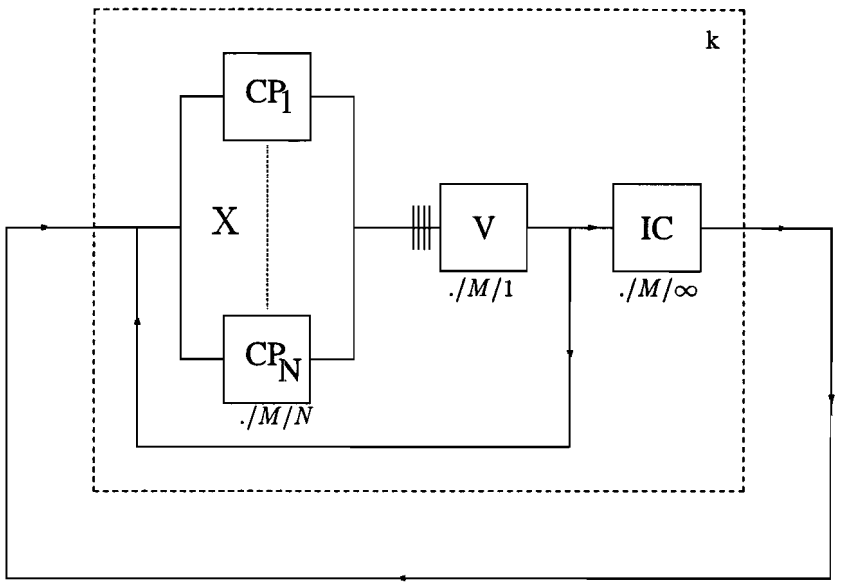


Figure 3: THE CLOSED SYSTEM

An exact analysis of the closed system with population k would require the state descriptor $(r_1, g_1, i_2, c_1, \ldots, c_{i_2})$, where r_1 (g_1) denotes the number of red (green) transactions at station 1, i_2 the number of transactions present at station 2, and $c_1, \ldots, c_{i_2}$ the colour of the transaction at position $1, \ldots, i_2$ in the queue of station 2. (Then $k - r_1 - g_1 - i_2$ transactions are present at station 3.) However the number of states then is of order $k^3 2^k$, which means already 1 million states for $k = 10$. Finding the steady-state probabilities for a non-productform network with this number of states is not feasible. Therefore an approximative analysis with a smaller state space is desirable. We investigated two approximative approaches for analysing the closed system with population k:

I State-dependent feedback with state description (r_1, g_1, r_2, g_2)

II Colouring transactions according to a Poisson process.

These approaches are discussed in the next subsections.

I State-Dependent Feedback with State Description (r_1, g_1, r_2, g_2)

If all service times at the stations in Figure 3 are taken exponential and if we assume (as already mentioned) that a committing transaction marks another transaction red with probability p, we can approximate the steady-state probability of having r_1 (g_1) red (green) transactions at the ample server, r_2 (g_2) red (green) transactions at the single server and thus, when k is the population of the closed system, $k - r_1 - g_1 - r_2 - g_2$ transactions at the infinite server. The probability that a transaction has to be rerun (the feedback probability) can then be modelled as being state dependent. It is approximated by $\frac{r_2}{r_2+g_2}$ if r_2 red and g_2 green transactions are present at the single validation server. As we will also see later, this approximation ignores the fact that most red transactions at station 2 are at the head of the

queue. The approximation thus underestimates the feedback probability. Note that there is always at least one green transaction in the box, so $r_1 + r_2 < k$.

We use the state description (r_1, g_1, r_2, g_2) and we write $B(n, m)$ for the binomial probability that after a validation of a green transaction m of the remaining n green transactions are marked red. So

$$B(n,m) := \binom{n}{m} p^m (1-p)^{n-m}.$$

Then the state transitions are:

- red service completions at X:
  $(r_1, g_1, r_2, g_2) \longrightarrow (r_1 - 1, g_1, r_2 + 1, g_2)$ with rate $r_1\mu_1$
- green service completions at X:
  $(r_1, g_1, r_2, g_2) \longrightarrow (r_1, g_1 - 1, r_2, g_2 + 1)$ with rate $g_1\mu_1$
- red service completions at V:
  $(r_1, g_1, r_2, g_2) \longrightarrow (r_1, g_1 + 1, r_2 - 1, g_2)$ with rate $\frac{r_2}{r_2+g_2}\mu_2$
- green service completions at V:
  $(r_1, g_1, r_2, g_2) \longrightarrow (r_1 + k_1, g_1 - k_1, r_2 + k_2, g_2 - 1 - k_2)$
  for $k_1 = 0, \ldots, g_1$ and $k_2 = 0, \ldots, g_2 - 1$ with rate $\frac{g_2}{r_2+g_2} B(g_1, k_1) B(g_2 - 1, k_2)\mu_2$
- service completions at IC:
  $(r_1, g_1, r_2, g_2) \longrightarrow (r_1, g_1 + 1, r_2, g_2)$ with rate $(k - (r_1 + g_1 + r_2 + g_2))\mu_3$.

The 'rate out of = rate into' principle leads to the following balance equation for the state-space interior, i.e. for the states with $r_1 > 0$, $g_1 > 0$, $r_2 > 0$, $g_2 > 0$ and $r_1 + g_1 + r_2 + g_2 < k$:

$$\begin{aligned}
&\pi(r_1, g_1, r_2, g_2)\,((r_1 + g_1)\mu_1 + \mu_2 + (k - (r_1 + g_1 + r_2 + g_2))\mu_3) \\
&\quad = (r_1 + 1)\mu_1 \pi(r_1 + 1, g_1, r_2 - 1, g_2) + (g_1 + 1)\mu_1 \pi(r_1, g_1 + 1, r_2, g_2 - 1) \\
&\qquad + \frac{r_2 + 1}{r_2 + g_2 + 1}\mu_2 \pi(r_1, g_1 - 1, r_2 + 1, g_2) \\
&\qquad + \sum_{k_1=0}^{r_1} \sum_{k_2=0}^{r_2} B(g_1 + k_1, k_1) B(g_2 + k_2, k_2) \frac{g_2 + 1 + k_2}{r_2 + g_2 + 1}\mu_2 \\
&\qquad\qquad \times \pi(r_1 - k_1, g_1 + k_1, r_2 - k_2, g_2 + 1 + k_2) \\
&\qquad + (k - (r_1 + g_1 - 1 + r_2 + g_2))\mu_3 \pi(r_1, g_1 - 1, r_2, g_2).
\end{aligned}$$

The balance equations for states at the state-space boundary are obtained in a similar way, but we omit them for reasons of brevity.

The equilibrium probabilities $\pi(r_1, g_1, r_2, g_2)$ can be computed by an iterative method like Gauss-Seidel that takes advantage of the sparsity of the transition matrix (see Van der Wal and Schweitzer [1987] for an implementation with lower and upper bounds on the probabilities).

We call a validation of a transaction successful if it has as outcome that the transaction need not be rerun. Define by $p_{suc}(k)$ the long-run average probability that a validation is

successful in a closed system with k customers. Denote by $n_c(k)$ and $n_v(k)$ respectively the mean number of commits and validations per time unit in the closed system with population k, and denote by $E[\text{Boxtime}(k)]$ the expected time an arbitrary transaction will spend in the closed system with population k. Let $\pi_k(r_1, g_1, r_2, g_2)$ be the steady-state probability for state (r_1, g_1, r_2, g_2) in a closed system with constant population k. Using $\pi_k(r_1, g_1, r_2, g_2)$, the performance measures of interest for a closed system with k customers can be approximated as follows:

$$\begin{aligned} n_c(k) &= \sum_{r_1+g_1+r_2+g_2<k} (k - r_1 - g_1 - r_2 - g_2)\mu_3\pi_k(r_1, g_1, r_2, g_2) \\ n_v(k) &= \sum_{r_2+g_2>0} \mu_2\pi_k(r_1, g_1, r_2, g_2) \\ p_{suc}(k) &= \frac{n_c(k)}{n_v(k)} \\ E[\text{Boxtime}(k)] &= \frac{k}{n_c(k)}. \end{aligned}$$

The throughput $\mu_{Box}(k)$ of the box (= average rate at which transactions leave the box) is approximated by $n_c(k)$.

Although the number of states is a lot smaller than in the exact approach with states $(r_1, g_1, i_2, c_1, \ldots, c_{i_2})$, it is still quite large, even with a small population k: the number of states (r_1, g_1, r_2, g_2) is equal to the number of ways in which k customers can be allocated to 5 categories (r_1, r_2, g_1, g_2 and the number at the infinite server), minus the one state with $r_1 + r_2 = k$. Thus the number of states is $\binom{k+4}{4} - 1$. For $k = 4$ this gives 69 states, for $k = 10$ already 1000 states and for $k = 20$: 11395 states.

Solving the balance equations for k near 20 is possible but requires very large memory sizes and long computation times. To avoid this problem, we also tried a model formulation with yet a smaller number of states and state description (i_1, i_2, r). There i_1 (i_2) is the number of transactions present at station X (V); the remaining $k - i_1 - i_2$ transactions are at station IC. Further, r is the total number of red transactions present at station 1 and 2 together. The probability of a rerun in that model is approximated by $\frac{r}{i_1+i_2}$. For k large, the number of states is about a factor $k/8$ smaller than the number of states in the model with state description (r_1, g_1, r_2, g_2). (When $k = 20$ there are 3290 states.) Unfortunately, numerical experiments showed that the model with state description (i_1, i_2, r) yields quite inaccurate approximations of $E[\text{Boxtime}(k)]$, compared to the approximations obtained from the model with states (r_1, g_1, r_2, g_2). The reason for this is the approximation $\frac{r}{i_1+i_2}$ for the rerun probability instead of $\frac{r_2}{r_2+g_2}$. In reality, the number of red transactions is not equally spread over station 1 and 2. At station 2 there are relatively more red transactions than at station 1. As it is inaccurate, we will not consider the model with state description (i_1, i_2, r) any further.

II Colouring Transactions according to a Poisson Process

A quite different approximation is obtained by making the following assumption.

Assumption:
Each transaction present at station 1 or in the queue of station 2 of the closed system is invalidated (coloured red) by other transactions according to a Poisson process.

This assumption may be reasonable because of the exponential service times. Denote the rate of the Poisson invalidation process by $\lambda(k)$ when there are k customers in the box. It seems fairly reasonable to assume, inspired by the Arrival Theorem for closed product-form queueing networks, that $\lambda(k) = n_c(k-1)p$ with $n_c(k-1)$ the mean number of commits in a closed system with population $k-1$.

From this Poisson invalidation assumption we have

$$P(\text{transaction still green after time } t \mid k \text{ in closed system}) = e^{-\lambda(k)t}.$$

The longer a transaction is present at station 1 or in the queue of station 2 of the closed system, the smaller is the probability that it is still green. This is reflected correctly by the approximation $e^{-\lambda(k)t}$. The validation of a transaction that is still green at the start of its validation is always successful. Hence the long-run average probability $p_{suc}(k)$ that a validation is successful when the population of the closed system is k, can be expressed as

$$p_{suc}(k) = E\left[e^{-\lambda(k)(S_1+W_2)}\right],$$

where S_1 is the response time of a transaction at station 1 (the ample server X) and W_2 is the waiting time of a transaction at station 2 (the validation server V) in an arbitrary transaction run.

To determine $p_{suc}(k)$ we adopt the following approximative approach. S_1 is exponential with rate μ_1, so the probability that the transaction is still green at the completion of its service at station 1 is equal to $\mu_1/(\mu_1 + \lambda(k))$. The probability that a transaction T is still green when it goes into service at station 2 given that it was green when it entered queue 2 is equal to $(\mu_2/(\mu_2 + \lambda(k)))^{i_2}$ when i_2 is the number of transactions in front of T at station 2 upon arrival. Hence

$$p_{suc}(k) = \sum_{i_2=0}^{k-1} P(\text{tr. finds } i_2 \text{ tr. at V} \mid k \text{ in closed system}) \left(\frac{\mu_1}{\mu_1 + \lambda(k)}\right) \left(\frac{\mu_2}{\mu_2 + \lambda(k)}\right)^{i_2}.$$

It remains to find an expression for $P(\text{tr. finds } i_2 \text{ tr. at V} \mid k \text{ in closed system})$ which is done below.

Assumption:
Given $p_{suc}(k)$, we assume that all transactions validating at station 2 have this fixed probability of success, independent of everything else in the queueing system. So we assume Markovian routing from station 2.

Result:
Given $p_{suc}(k)$, the closed network is of product form. Thus the equilibrium distribution is

known and the Arrival Theorem holds.

Under the above assumption we have a closed queueing network with 3 stations (X, V and IC), k single-class customers and a fixed feedback probability $1 - p_{suc}(k)$. Taking as state descriptor (i_1, i_2, i_3) (where i_k indicates the number of customers at station k), the equilibrium distribution $\pi_k(i_1, i_2, i_3)$ is:

$$\pi_k(i_1, i_2, i_3) = C \frac{1}{i_1!} \left(\frac{1}{\mu_1}\right)^{i_1} \left(\frac{1}{\mu_2}\right)^{i_2} \frac{1}{i_3!} \left(\frac{p_{suc}(k)}{\mu_3}\right)^{i_3}. \tag{1}$$

Here C is the normalising constant. For this product-form queueing network it follows from the Arrival Theorem that

$$P(\text{tr. finds } i_2 \text{ tr. at V} \mid k \text{ in closed system}) = \sum_{i_1=0}^{k-1-i_2} \pi_{k-1}(i_1, i_2, k-1-i_1-i_2),$$

with $\pi_{k-1}(i_1, i_2, k-1-i_1-i_2)$ computable in the same way as (1). Thus

$$p_{suc}(k) = \sum_{i_2=0}^{k-1} \sum_{i_1=0}^{k-1-i_2} \pi_{k-1}(i_1, i_2, k-1-i_1-i_2) \left(\frac{\mu_1}{\mu_1 + \lambda(k)}\right) \left(\frac{\mu_2}{\mu_2 + \lambda(k)}\right)^{i_2}. \tag{2}$$

Further,

$$n_c(k) = \sum_{i_1+i_2<k} (k - i_1 - i_2)\mu_3 \pi_k(i_1, i_2, k - i_1 - i_2).$$

Using

$$\lambda(k) = n_c(k-1)p, \tag{3}$$

we can compute the value of $p_{suc}(k)$ by recursion. Starting with $p_{suc}(1) = 1$, $\lambda(2)$ follows from (3) and $p_{suc}(2)$ from (2). Doing this $k-1$ times yields the value of $p_{suc}(k)$.

The mean response time of a transaction in the box is approximated by

$$E[\text{Boxtime}(k)] = \frac{k}{n_c(k)}.$$

Then the throughput of the box, $\mu_{Box}(k)$, that will be used in the next section to approximate the expected total response time of a transaction in the database, is given by $n_c(k)$.

The advantage of this second approach for estimating $\mu_{Box}(k)$ is that the equilibrium probabilities used are given by the explicit formula (1). No system of equations has to be solved in order to compute the equilibrium probabilities: this method can handle any value of k, no matter how big k is. It only takes k steps to find the approximating value for $p_{suc}(k)$.

The approximating equation (2) for $p_{suc}(k)$ can be slightly improved. We approximated the probability that given a transaction T is green when it enters queue 2 it is still green when it goes into service at station 2 by $(\mu_2/(\mu_2 + \lambda(k)))^{i_2}$ when i_2 transactions are ahead of T at station 2 upon arrival. Here the assumption of a Poisson invalidation process was used. However, when a transaction T at the validation station has i_2 transactions ahead of it, the

probability that T is not invalidated by any of these i_2 transactions can also be approximated by

$$(1 - p \cdot p_{suc}(k-1))^{i_2}.$$

Numerical experiments showed that this approximation is slightly better than the former. Therefore the approximation for $p_{suc}(k)$ used in approach **II** throughout the remainder of this paper is

$$p_{suc}(k) = \sum_{i_2=0}^{k-1} \sum_{i_1=0}^{k-1-i_2} \pi_{k-1}(i_1, i_2, k-1-i_1-i_2) \left(\frac{\mu_1}{\mu_1 + \lambda(k)}\right) (1 - p \cdot p_{suc}(k-1))^{i_2}.$$

4.2 Tying Things Together

As already mentioned at the beginning of section 4 we use a decomposition-aggregation approach for approximating the (mean) response time of a transaction on its path through the database system. In this section we use the results of the subsystem (the box) as input for the complete system. Figure 4 illustrates the aggregation approach.

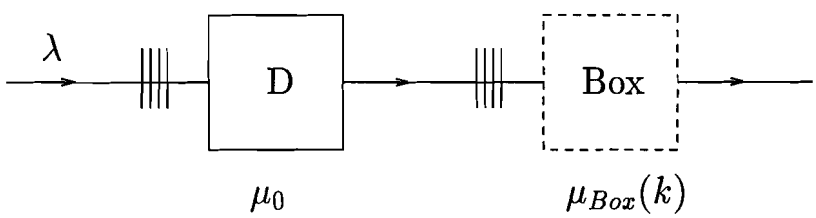


Figure 4: AGGREGATION

We now have two stations, D and Box. Box is considered as a FCFS exponential service station with service rate $\mu_{Box}(k)$ when the number of customers in front of plus inside the box is $k < N$, and $\mu_{Box}(N)$ when the total number of customers at the box is bigger than or equal to N. Denoting the state of the aggregate system by (j_1, j_2) when j_1 and j_2 transactions are present at station 1 (the single server D) and station 2 (Box) respectively, the state transitions lead to the following balance equation for the interior of the state space:

$$\begin{aligned}&\pi(j_1, j_2)(\lambda + \mu_0 + \mu_{Box}(\min\{j_2, N\})) \\ &\quad = \lambda\pi(j_1 - 1, j_2) + \mu_0\pi(j_1 + 1, j_2 - 1) + \mu_{Box}(\min\{j_2 + 1, N\})\pi(j_1, j_2 + 1).\end{aligned}$$

In fact, the steady-state probabilities are given by a product form.
Denote by $E[L_{Box}]$ the long-run average number of customers present in front of plus inside Box. Then

$$E[L_{Box}] = \sum_{j_2} j_2 \pi(j_2),$$

where $\pi(j_2)$ is the marginal probability of having j_2 transactions present at station 2, and is given by

$$\pi(j_2) = C\lambda^{j_2} \prod_{j=1}^{j_2} \frac{1}{\mu_{Box}(\min\{j, N\})},$$

with C a normalising constant. Using Little's law, the total mean response time is approximated by

$$E[S] = \frac{1}{\mu_0 - \lambda} + \frac{1}{\lambda} E[L_{Box}].$$

5 Numerical Results

The queueing model of the database system as shown in Figure 2 is not an exact representation of the database system. For instance, the queueing model does not use exact information about which data are needed by which transaction, while in reality the database with OCC implementation does use this information to determine whether a transaction has to be rerun or not. As already mentioned, a comparison of (a simulation of) the real database system with a simulation of the simpler queueing system that makes use of red and green transactions and probabilistic conflicts should reveal whether the queueing representation is accurate enough.

Unfortunately, the queueing model itself cannot be analysed exactly. This is mainly caused by the blocking that occurs in front of the dashed box when there are N transactions present in the box, but also by the feedback probability that is not constant.

In this section we compare the approximations for the queueing model as derived in section 4 with a simulation of the queueing model. For every transaction the simulation program keeps a record of its colour and its time spent at the stations D, X, V, and IC (= I + C). In the simulation, the colour of a transaction in the box can change from green to red when another transaction enters its commit phase after a successful validation. The validation of a red transaction is unsuccessful and sends the transaction back as a green transaction to the CPU, a green transaction always validates successfully and colours each transaction present red with probability p. Every time a transaction is rerun, a fresh execution time is drawn from the exponential distribution.

Simulation programs were built both for the complete queueing network of Figure 2 and for the closed queueing network of Figure 3. For a heavy-loaded database system, the simulation results for the expected time a transaction spends in the box in both models coincide.

Optimistic concurrency control is only useful for low values of the conflict probability p, say $p \leq 0.2$. If p is too high, the number of reruns per transaction will be large and a lot of computing resources are wasted because the probability that a validation is successful is very small. Moreover, not only the time a transaction spends inside the dashed box of Figure 2 will increase dramatically as p increases, but (because of this) also the length of the queue of waiting transactions in front of the box will grow, resulting in long waiting times. In cases where p is high, a locking scheme usually performs better (see Huang *et al.* [1991]). Hence the numerical experiments reported below were performed with $p \leq 0.2$.

The conflict probability p is computed as the probability that two transactions have at least one data-item in common. To give an idea of p for various database sizes, we constructed Table 1. If the database consists of L data-items and every transaction uses exactly ℓ data-items uniformly distributed on the total of L items, the probability p that two transactions have one or more data-items in common equals

$$p = 1 - \frac{\binom{\ell}{0}\binom{L-\ell}{\ell}}{\binom{L}{\ell}} = 1 - \frac{[(L-\ell)!]^2}{L!(L-2\ell)!}.$$

Observe in Table 1 that p does not solely depend on the fraction ℓ/L.

ℓ	1	1	5	5	10	10	10	100	100	1000
L	100	10000	100	10000	100	1000	10000	10000	10^6	10^6
p	0.0100	0.0001	0.2304	0.0025	0.6695	0.0960	0.0100	0.6358	0.0100	0.6325

Table 1: CONFLICT PROBABILITY p

In section 5.1 we compare the analytic approaches for the closed system with a simulation of the closed system. In section 5.2 we compare the aggregation results of the analyses with a simulation of the complete queueing system.

5.1 The Closed System

We consider the following cases:

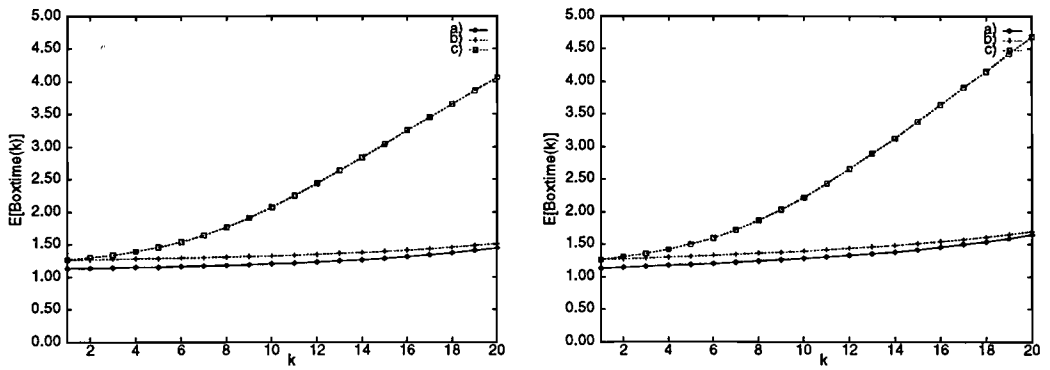
a) $\mu_1 = 1, \mu_2 = 15, \mu_3 = 15$
b) $\mu_1 = 1, \mu_2 = 15, \mu_3 = 5$
c) $\mu_1 = 1, \mu_2 = 5, \mu_3 = 15.$

In case a) we assume that the validation phase of a transaction is on average 15 times faster than the execution phase, and that the initialisation and the commit phase together take as much time as the validation phase. In case b), the validation is 3 times as fast as the initialisation and commit phase together. Finally, in case c), the validation phase is only 5 times as fast as the execution phase so for a closed system with both the number of CPUs N and the population k bigger than 5, the validation station may become a bottle-neck. We varied the population k of the closed system from 1 to 20, and computed for both the approximative methods **I** and **II** the values of the success probability $p_{suc}(k)$, the throughput $\mu_{Box}(k)$ and the expected time a transaction spends in the box, $E[\text{Boxtime}(k)]$. All this was done for various values of the conflict probability p, namely $p = 0.001$, $p = 0.01$, $p = 0.1$ and $p = 0.2$. Below, the results of the approximative methods are compared with results obtained via a simulation of the closed system. With 'simulation result' we refer to the midpoint of a 95% confidence interval for $E[\text{Boxtime}(k)]$, which in all cases has a width smaller than 1 percent of the midpoint value.

For $p = 0.001$, we found in all cases a) to c) and for all k, that the approximations for $E[\text{Boxtime}(k)]$ differ less than 1% from the simulation result. To give an idea of the value of $E[\text{Boxtime}(k)]$ for varying k, the simulation results for cases a) to c) are depicted in Figure 5(a). We see that the time needed for validation can, even for very small values of p, have a tremendous effect on the average time a transaction spends in the box. This shows that neglecting the validation time as was done in previous studies (Menascé and Nakanishi [1982], Morris and Wong [1985], Kleinrock and Mehović [1992], and Yu *et al.* [1993]) may not be justified.

For $p = 0.01$, again the approximative methods perform well compared to simulation. However, method **II** is slightly better than **I**. The maximum difference of both methods compared to simulation is 1%. We give a graph of the simulated values of $E[\text{Boxtime}(k)]$ in Figure

5(b), so that they can be compared with Figure 5(a).



(a) $p = 0.001$ (b) $p = 0.01$

Figure 5: SIMULATION RESULTS FOR CLOSED SYSTEM

For $p = 0.1$ and $p = 0.2$, we plotted the results for $E[\text{Boxtime}(k)]$ of both **I**, **II**, and simulation together in one graph. This was done for each of the cases a) to c), and led to the Figures 6(a) to (c) as given below. It is clear from the figures that the simple heuristic approach **II** performs excellently compared to simulation; the lines representing approach **II** and the simulation results almost overlap in all three cases. Method **I** is only good in case b). For cases a) and c) with $N > 8$, method **I** deviates significantly from the simulation. An explanation for this lies in the fact that the rerun probability is not accurately approximated by $\frac{r_2}{r_2+g_2}$ anymore when the mean number of transactions present at the validation server becomes larger. In reality, red transactions are not equally spread over the positions in the queue of station 2. Actually, most red transactions at station 2 are at the head of the queue, and most green transactions are at the tail of the queue. Hence, method **I** underestimates the probability of a rerun and the expected time a transaction spends in the box. In case a) and c), the mean queue length at station 2 is larger than in case b). When the queue length at station 2 is larger, the error made by still taking $\frac{r_2}{r_2+g_2}$ as the probability of a rerun is larger. This explains that for case a) and c), method **I** yields a less accurate approximation for the rerun probability — and thus for $E[\text{Boxtime}(k)]$ — than for case b).

Another remarkable aspect of Figure 6 is, that for k large there is no significant difference between the simulation results of cases a) and b). One would expect that the mean time spent in the box in case b) would be about 0.13 larger than in case a), because the mean commit time in case b) differs that amount from the mean commit time in case a). However also this phenomenon can be explained: as in case b) the mean commit time is larger, the average queue length at station 2 is smaller. This leads to a smaller cycletime per transaction and thus to a smaller $E[\text{Boxtime}(k)]$. Apparently the extra commit time in case b) is compensated by a smaller waiting time at station 2.

To also give an impression of the values of $p_{suc}(k)$ and $\mu_{Box}(k)$, Table 2 shows those values together with $E[\text{Boxtime}(k)]$ for case c) with $p = 0.1$. As the throughput is maximal for $k = 8$, we can conclude from the table that having more than 8 CPUs is useless in case c) with $p = 0.1$. Actually, for every choice of system parameters there is a number k^* of CPUs

for which the throughput is maximal.

k	1	2	3	4	5	6	7	8	9	10
$p_{suc}(k)$	1.000	0.918	0.855	0.803	0.759	0.722	0.688	0.658	0.629	0.603
$\mu_{Box}(k)$	0.790	1.420	1.926	2.329	2.642	2.854	2.978	3.018	3.004	2.945
E[Boxtime(k)]	1.266	1.408	1.557	1.717	1.892	2.103	2.350	2.651	2.996	3.396

k	11	12	13	14	15	16	17	18	19	20
$p_{suc}(k)$	0.579	0.558	0.538	0.521	0.504	0.488	0.474	0.461	0.449	0.438
$\mu_{Box}(k)$	2.862	2.777	2.684	2.601	2.518	2.440	2.367	2.306	2.242	2.188
E[Boxtime(k)]	3.844	4.322	4.844	5.383	5.957	6.558	7.181	7.806	8.474	9.142

Table 2: SIMULATION RESULTS FOR $\mu_1 = 1$, $\mu_2 = 5$, $\mu_3 = 15$, WITH $p = 0.1$

With respect to the performance of the approximative methods **I** and **II** for the closed system, we can conclude for all cases a) to c) that

- as k or p increases, **I** diverges from the simulation results whereas **II** stays fairly close to the simulation,
- for small p **II** (slightly) overestimates E[Boxtime(k)], for high p **II** (slightly) underestimates E[Boxtime(k)]. **I** always underestimates E[Boxtime(k)].

5.2 The Complete System

In the previous section we saw that approach **II** yields a very good approximation for E[Boxtime(k)] in a closed system with population k. This good performance holds for a wide range of choices for p and the relative speeds of the execution, validation and commit phases. Now we investigate the performance of the approximative approaches **I** and **II** for the complete system. As described in section 4.2, we use the throughput results obtained by **I** and **II** for the closed system as input for the complete system.

Input parameters

Again we look at the three cases

a) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 15$
b) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 5$
c) $\mu_1 = 1$, $\mu_2 = 5$, $\mu_3 = 15$.

The performance measure of interest is $E[S]$, the average total response time of a transaction. The conflict probability p is varied from 0.01, 0.1 to 0.2 (we drop the case $p = 0.001$ because there conflicts are too rare). The speed of the dedicated CPU D is set at $\mu_0 = 40$. We approximate and simulate $E[S]$ for database systems with the number of CPUs N varied from 1 to 20.

For each combination of the input parameters μ_1, μ_2, μ_3, and p, the arrival rate λ is chosen such that the ratio $\lambda/\mu_{Box}(8)$ is between 0.6 and 0.8. Here $\mu_{Box}(8)$ is the simulated value of the throughput of the closed system with 8 CPUs and a fixed number of 8 transactions. The throughput $\mu_{Box}(k)$ for a closed system with k transactions is first increasing in k, but, due

to more conflicts, decreases in k for k large (attaining its maximum value for k^* somewhere in between). A system with $N = 8$ is taken as a reference point for choosing λ because in case c), k^* is around 8. Hence, by choosing λ such that $\lambda/\mu_{Box}(8)$ is around 0.7, the corresponding database systems with the same arrival rate but a smaller number of CPUs (say 4 or 5) are overloaded. Systems with the same arrival rate but a larger number of CPUs can also become overloaded, resulting in a large $E[S]$ as we will see below.

The advantage of keeping λ fixed while varying N is that it shows the effect of having more CPUs on the response time. This is an important issue in the design of real-time database systems.

Results

For $p = 0.01$, the results are given in Figure 7. We see that both approximations match the simulation results. Only in case c) for $N = 5$ and $N = 6$, the approximations differ more than 10% from the simulation. However this is due to the error made in the decomposition-aggregation step. The throughput $\mu_{Box}(k)$ of the closed system was excellently approximated by **I** and **II** for all k, so the error in approximating $E[S]$ must have been caused by treating the box in the decomposition-aggregation step as a FCFS service station with *exponential* service rate $\mu_{Box}(k)$. Apparently, when the number of CPUs N is small, the error introduced by this 'exponentiality'-assumption is larger than when N is large. This may also have to do with the fact that for N small, hardly any waiting occurs at the validation station.

For $p = 0.1$, Figure 8 shows the results. Again for the heavy-loaded cases of N small, the decomposition-aggregation step introduces an error. For N between 6 and 20 for cases a) and b), approaches **I** and **II** give very good approximations of $E[S]$. In case c) the approximations are good for N between 6 and 13. For $N > 13$, the system becomes overloaded which is recognized by approach **II** but not by approach **I**. This is because **I** overestimates $\mu_{Box}(k)$ for k large, so that according to **I** the system is still stable (not overloaded) for N between 14 and 20. Anyway, considering $N > 8$ in case c) is useless, since extra CPUs only lower the throughput.

For $p = 0.2$, the results are plotted in Figure 9. Here the same remarks apply as in the case with $p = 0.1$.

The overall conclusion with respect to the approximations **I** and **II** for $E[S]$ is that they give accurate results for systems that are not heavy-loaded. Method **II** is at least as good as **I**, so is preferred because of its computational simplicity. Further, depending on the system parameters, there is a number N^* of CPUs the system should have in order to minimize the average response time. Adding extra CPUs beyond N^* may have a dramatic effect on the response times.

6 Conclusions

In order to evaluate the performance of optimistic concurrency control in a multi-processor shared-memory database, we modelled the real-time database system as a queueing network. Since the queueing network is not of product form, approximations are needed to compute the average response time $E[S]$ of a transaction in the system. We studied two approaches. The first one requires solving a large system of (balance) equations in order to get the steady-state probabilities of the queueing network. The second one is much less elaborate and is based on the assumption that transactions leave the system according to a Poisson process. Although

the assumption is not generally true, it does resolve in a fairly simple iterative algorithm giving a good approximation of $E[S]$.

The numerical experiments showed that method **II** is an excellent method for approximating the time spent by a transaction in a system with a fixed number of transactions. For such a closed system, method II is better than the time-consuming method **I**. For the open system, method **II** performes well for systems that are not heavy-loaded. Method **I** also gives good approximations of $E[S]$ but **II** is preferred because of its simplicity and its negligible computation times. Moreover, for heavy-loaded systems with a large number of CPUs, method **II** produces a much better approximation for the average response time than method **I**.

Future research topics include the extension of the model to non-homogeneous data-access (i.e. the occurrence of popular and non-popular data-items), variable transaction sizes and variable transaction deadlines (i.e. transactions with different and dynamically changing priorities). Also, the analysis can be adapted for nonexponential execution and commit times. A very important research subject with respect to performance modelling of real-time databases that has — to our knowledge — not yet emerged in literature, is the development of an approximation for the distribution of the response time of transactions.

References

Agrawal, R., M.J. Carey, and M. Livny [1987], Concurrency control performance modeling: alternatives and implications, *ACM Transactions on Database Systems* **12**, 609–654.

Baskett, F., K.M. Chandy, R.R. Muntz, and F. Palacios-Gomez [1975], Open, closed and mixed networks of queues with different classes of customers, *Journal of the ACM* **22**, 248–260.

Huang, J., J.A. Stankovic, K. Ramamritham, and D. Towsley [1991], Experimental evaluation of real-time optimistic concurrency control schemes, *Proceedings of the 17th International Conference on Very Large Data Bases (eds. G.M. Lohman, A. Sernadas, and R. Camps)*, pp. 35–46

Kleinrock, L. and F. Mehović [1992], Poisson winner queues, *Performance Evaluation* **14**, 79–101.

Kung, H. and J. Robinson [1981], On optimistic methods for concurrency control, *ACM Transactions on Database Systems* **6**, 213–226.

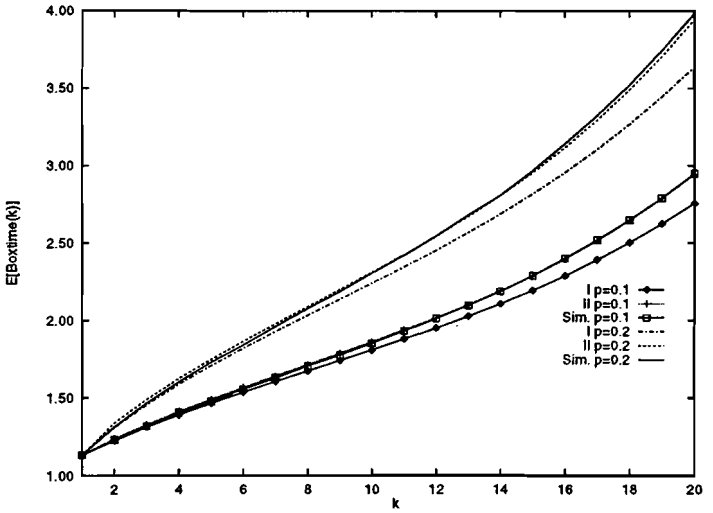
Menascé, D.A. and T. Nakanishi [1982], Optimistic versus pessimistic concurrency control mechanisms in database management systems, *Information Systems* **7**, 13–27.

Morris, R.J.T. and W.S. Wong [1985], Performance analysis of locking and OCC algorithms, *Performance Evaluation* **5**, 105–118.

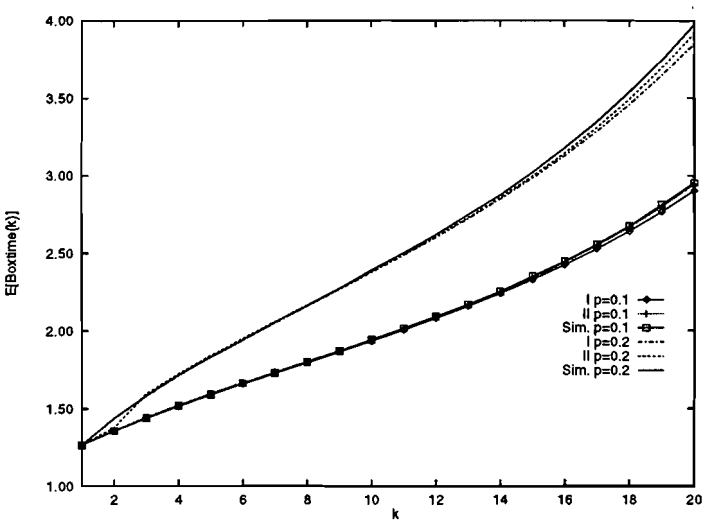
Papadimitriou, C. [1986], *The Theory of Database Concurrency Control*, Computer Science Press, Rockville, Md..

Van der Wal, J. and P.J. Schweitzer [1987], Iterative bounds on the equilibrium distribution of a finite Markov chain, *Probability in the Engineering and Informational Sciences* **1**, 117–131.

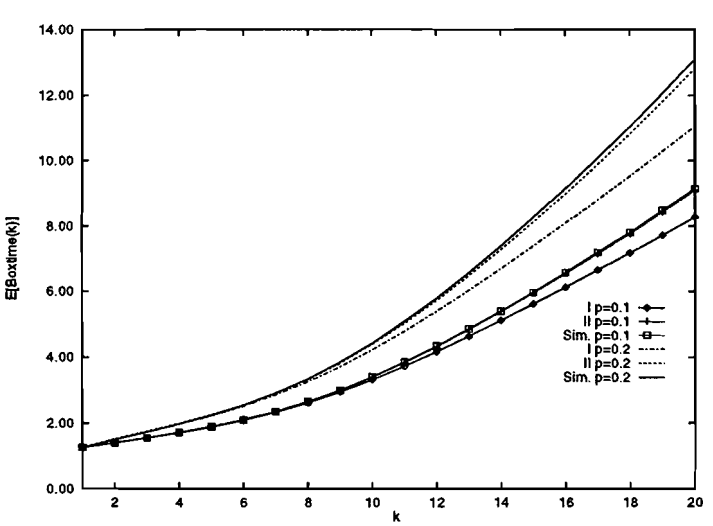
Yu, P.S., D.M. Dias, and S.S. Lavenberg [1993], On the analytical modeling of database concurrency control, *Journal of the ACM* **40**, 831–872.



(a) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 15$

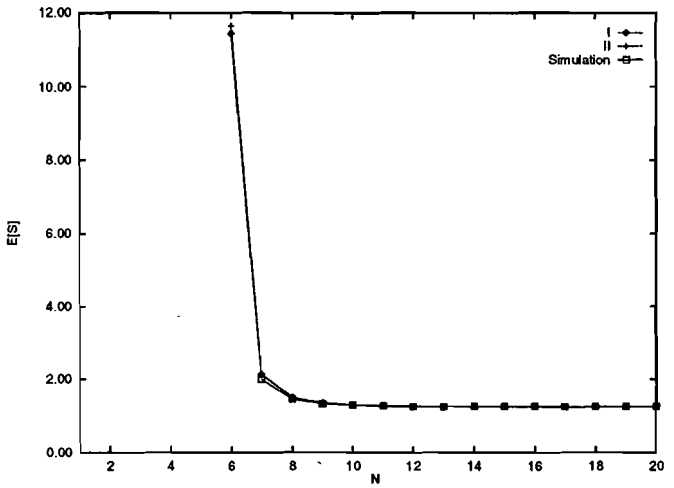


(b) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 5$

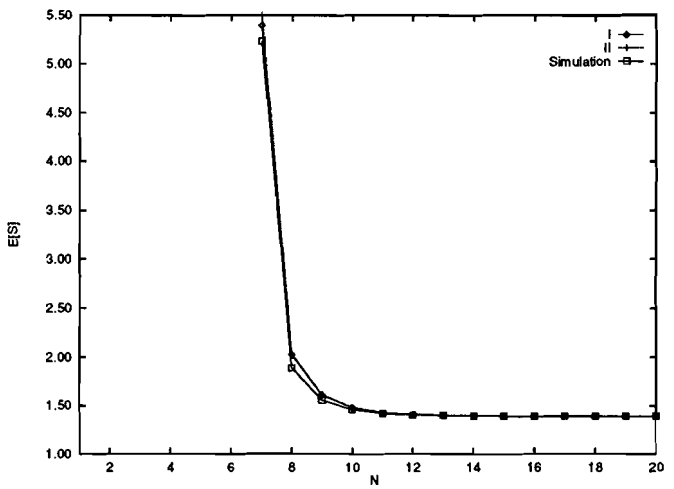


(c) $\mu_1 = 1$, $\mu_2 = 5$, $\mu_3 = 15$

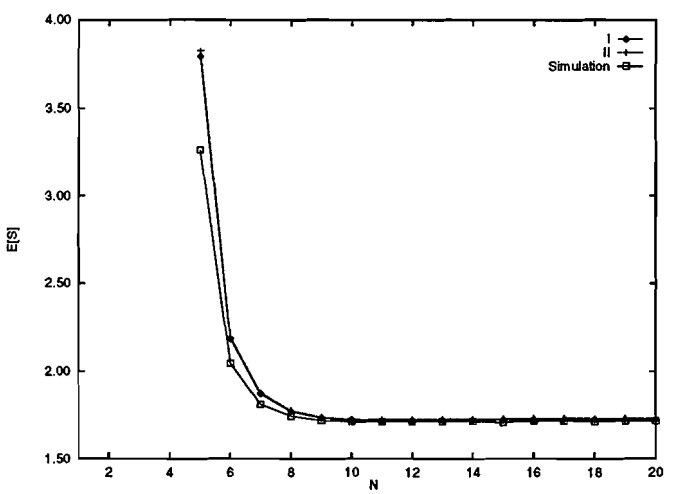
Figure 6: CLOSED SYSTEM RESULTS FOR $E[\text{BOXTIME}(k)]$ WITH $p = 0.1$ AND $p = 0.2$



(a) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 15$, $\lambda = 5$

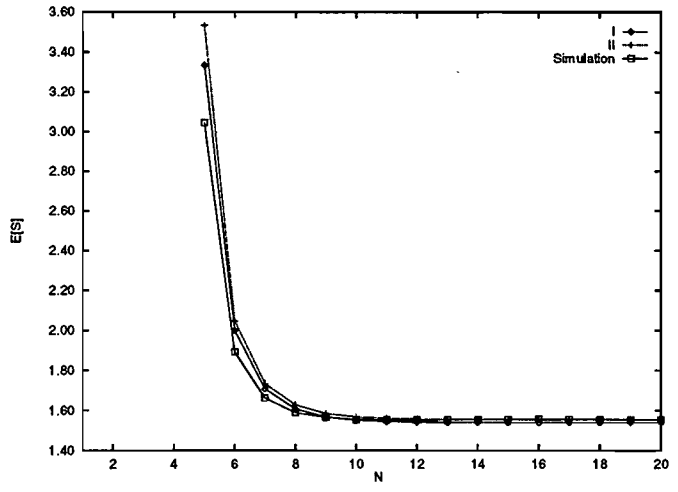


(b) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 5$, $\lambda = 5$

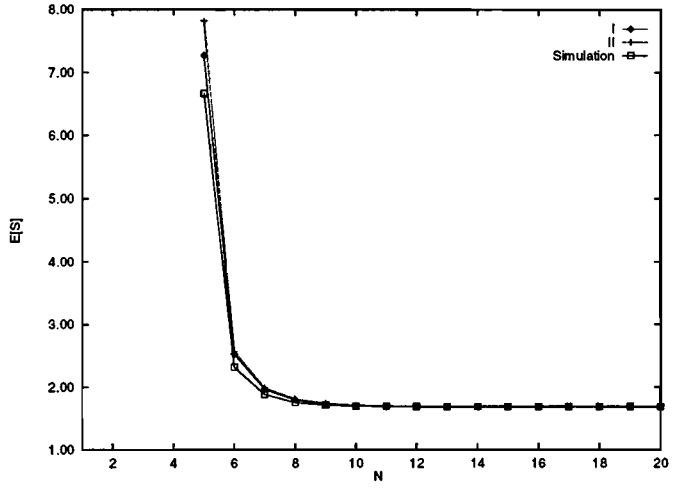


(c) $\mu_1 = 1$, $\mu_2 = 5$, $\mu_3 = 15$, $\lambda = 3$

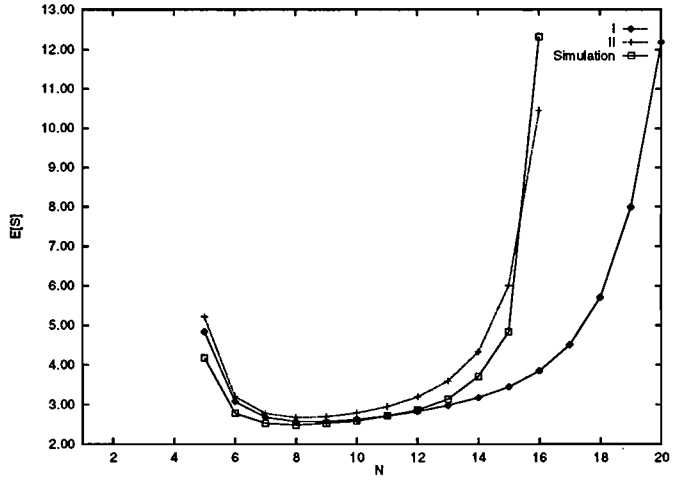
Figure 7: APPROXIMATIVE VERSUS SIMULATION RESULTS FOR $E[S]$ WHEN $p = 0.01$



(a) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 15$, $\lambda = 3$

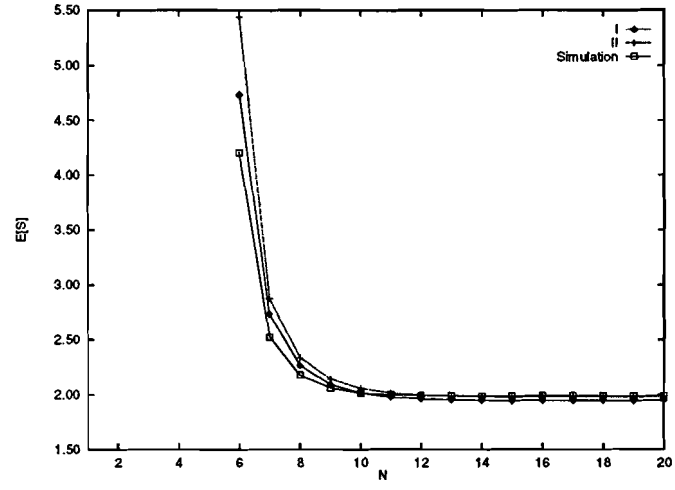


(b) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 5$, $\lambda = 3$

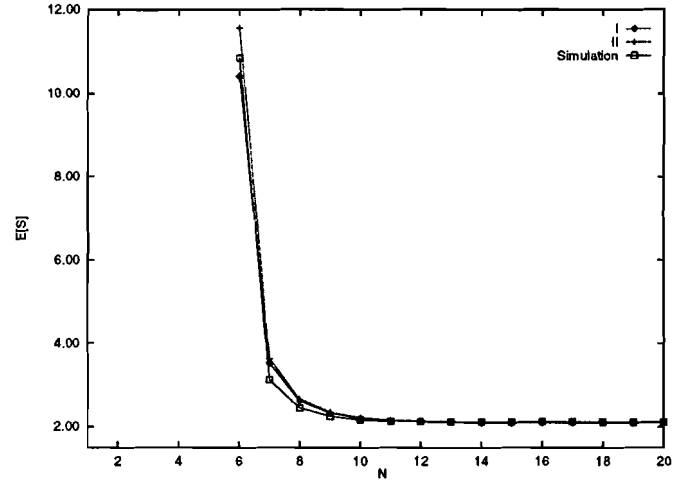


(c) $\mu_1 = 1$, $\mu_2 = 5$, $\mu_3 = 15$, $\lambda = 2.4$

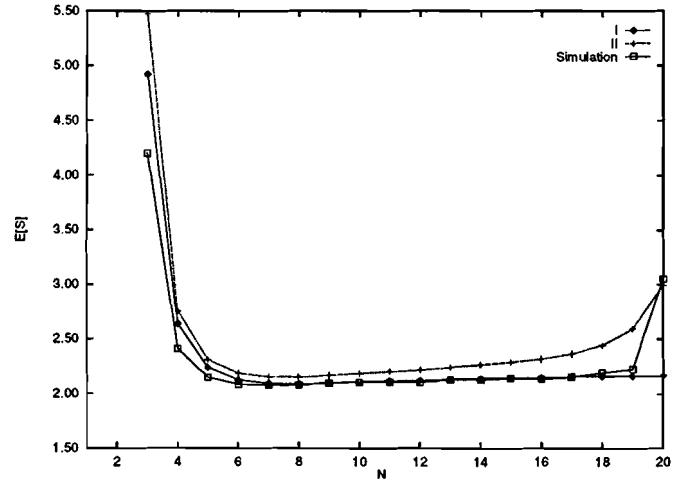
Figure 8: APPROXIMATIVE VERSUS SIMULATION RESULTS FOR $E[S]$ WHEN $p = 0.1$



(a) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 15$, $\lambda = 3$



(b) $\mu_1 = 1$, $\mu_2 = 15$, $\mu_3 = 5$, $\lambda = 3$



(c) $\mu_1 = 1$, $\mu_2 = 5$, $\mu_3 = 15$, $\lambda = 1.5$

Figure 9: APPROXIMATIVE VERSUS SIMULATION RESULTS FOR $E[S]$ WHEN $p = 0.2$