The response-time distribution in a real-time database with optimistic concurrency control and exponential execution times

Eindhoven University of Technology

Department of Mathematics and Computing Science

Memorandum COSOR 97-03

The Response-Time Distribution in a Real-Time Database with Optimistic Concurrency Control and Exponential Execution Times

S.A.E. Sassen
J. van der Wal

Eindhoven, February 1997
The Netherlands

The Response-Time Distribution in a Real-Time Database with Optimistic Concurrency Control and Exponential Execution Times*

Simone Sassen and Jan van der Wal

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

ABSTRACT

For a real-time shared-memory database with optimistic concurrency control, an approximation for the transaction response-time distribution is obtained. The model assumes that transactions arrive at the database according to a Poisson process, that every transaction takes an exponential execution time and uses an equal number of data-items uniformly chosen, and that the multiprogramming level is bounded. The analysis is based on a decomposition approach: results for the closed system with a fixed number of transactions are used to derive the response-time distribution in the open system with Poisson arrivals. Numerical experiments that compare analysis with simulation indicate that the approximation for the throughput and the response-time distribution works well for the closed system. For the open system the approximation for the response-time distribution is useful if the load is not too high.

1. INTRODUCTION

Real-time databases combine the requirements of both databases and real-time systems. In a database, transactions (database requests) should preserve database consistency. Subject to this consistency requirement, the transaction throughput of the database should be maximized. In a real-time system, the main requirement is timeliness, i.e., transactions must be executed before their deadlines. Soft real-time systems are allowed to miss some deadlines when the system is overloaded, but at least a certain fraction of the transactions should meet some prescribed deadline. In a real-time database, both consistency and timeliness are important. We investigate soft real-time databases in this paper and are interested in the probability that a transaction meets its deadline.

To benefit from the increase in CPU power that parallel computer architectures offer, transactions on databases should be executed concurrently. However, concurrent execution can destroy database consistency if conflicting transactions are incorrectly scheduled. Two transactions conflict if they access the same data-item, at least one of them with the intention to write. Concurrency control schemes govern the simultaneous execution of transactions such that overall cor-

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

rectness of the database is maintained (see e.g. [7]). The two main concurrency control schemes are locking and optimistic concurrency control.

Under the locking scheme, an executing transaction holds locks on all data-items it needs for execution, thus introducing lock waits for transactions that conflict with it. Consistency is guaranteed, however chains of lock waits can lead to long transaction response times.

When the conflict probability is low, it can be advantageous to use the optimistic concurrency control (OCC) scheme proposed in [4]. Under OCC, all CPUs can be used for transaction processing at the same time. Each transaction is processed in three phases: an execution phase, a validation phase and a commit phase. In the execution phase a transaction T accesses any data-item it needs for execution, regardless of the number of transactions already using that data-item. In the validation phase, all items used by T are checked for conflicts. If a conflict has occurred with a transaction that committed after T started, T must be rerun. If no conflicts occurred, T completes successfully and enters the commit phase, where the data-items used by T are updated.

There have been numerous performance studies of locking and optimistic concurrency control: simulation studies as well as analytical models. Our primary interest is in analytical performance studies. Therefore, the only simulation study we mention here is [1], which gives an extensive treatment of the influence the modeling assumptions with respect to resources and transaction behavior can have on the outcome of a performance study. A collection of analytical models for locking is [8]. Analytical models for OCC are [3] and [5]. Actually, [5] compares the performance of locking and OCC. Other analytical comparative studies are [6] and [10].

All analytical studies cited above are stochastic analyses that only consider *average* system performance. Typical performance measures studied are the throughput, the mean response time, the probability of a lock wait, and the average number of restarts needed for a transaction under OCC. To our knowledge, except for the study [2] for a variant of locking, no analytical performance studies of real-time databases exist that address the variance or the *distribution* of the response time. An approximation for the response-time distribution is needed to estimate the probability that a transaction meets its deadline, or the percentage of transactions that miss their deadline; these quantities cannot be derived from the throughput alone.

In this paper we approximate the response-time distribution in a multi-processor shared-memory database with optimistic concurrency control. The approach is based on the throughput analysis [6] of Morris and Wong. The model is explained in Section 2. In Section 3, an approximation is derived for the response-time distribution in a closed system with a fixed number of transactions, and its results are compared with simulation results. In Section 4 we analyze the response-time distribution in an open system with Poisson arrivals and test our approximation against simulation. Section 5 contains some concluding remarks.

2. THE MODEL

We model OCC in a shared-memory environment with N parallel CPUs as a multi-server queueing system with feedback, see Figure 1 for an illustration. In the dashed area, which represents the N CPUs, at most N transactions can be present. Each transaction is handled by one CPU and either leaves the system (after a successful execution), or is rerun (in case of a conflict). We assume that the time needed for one execution plus validation of a transaction is exponentially distributed with parameter μ. Further, it is assumed that the commit phase takes negligible time compared to execution plus validation, and that the validation can be efficiently implemented such that parallel validation is possible. The assumption that commit takes negligible time compared to execution plus validation is reasonable, since there are no disks attached to the system and all data-items are in main memory. Transactions arrive at the database according to a Poisson process with rate λ. An arriving transaction that finds all CPUs busy joins the queue. As soon as a CPU is freed by a departing transaction, the transaction that is first in queue is taken into execution. We also refer to execution plus validation as one transaction run.

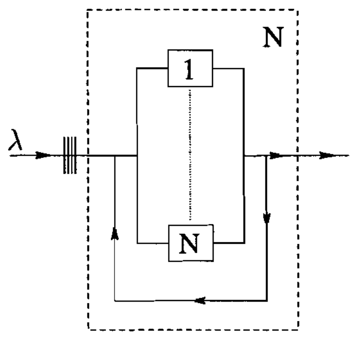


Figure 1. Queueing model of the open system

The model described is largely the same as the model of Morris and Wong [6]. With regard to transaction behavior, in accordance with Morris and Wong we assume that

- each transaction uses exactly a data-items, uniformly picked from the total set of d items,
- the time needed for a rerun of a transaction exactly equals the time of the first run, and
- all transactions write the data-items accessed, so there are no read-only transactions.

In light of these assumptions, we define b as the probability that two arbitrary transactions conflict. Then b can be computed from $b = 1 - \binom{d-a}{a} \Big/ \binom{d}{a}$. In the remainder of this paper, we will characterize the occurrence of conflicts only in terms of b.

We are aware that some of the assumptions made are not realistic (e.g., the exponentially-distributed execution time) or very restrictive (e.g., the uniform data-access pattern). However, for our first attempt at analyzing the response-time distribution of a real-time database with OCC, all these assumptions are needed. Once this model has been analyzed satisfactorily, the analysis will be extended such that some restrictive assumptions can be dropped.

The queueing model of Figure 1 is no standard feedback model. The probability that a transaction T must be rerun is not fixed, but depends on the number of transactions that departed (committed) during the execution of T. The number of departures during T's execution depends on the length of T's execution and on the number of concurrently executing transactions.

We analyze the model using a decomposition approach. First we approximate the response-time distribution in a so-called closed system with a constant population (multiprogramming level) of k transactions ($k \leq N$). The population is kept constant at k by admitting a new transaction to the system as soon as another transaction has committed. Next we consider the open system with Poisson arrivals and approximate the distribution of the response time using the results of the closed system. Such a decomposition approach has become fairly standard for computing the throughput and average response time of complicated queueing systems. We will investigate for the model of Figure 1 if the decomposition approach also gives accurate results if it is used to approximate the entire distribution of the response time.

For both the closed and the open system, numerical results from the analysis are compared with simulation results. In the simulation programs, the probability that a committing transaction conflicts with a transaction in execution is taken equal to b, in agreement with the above model description. Thus, no actual lists of data-items are used in the simulations. Also, the simulations take the time needed for a rerun of a transaction exactly equal to the time of the first transaction run.

3. RESPONSE-TIME DISTRIBUTION IN THE CLOSED SYSTEM

We consider the closed system of Figure 2, for a fixed multiprogramming level k. After a transaction run, a transaction can either be fed back for a rerun (requiring the same amount of time as the previous run), or leave. The moment a transaction leaves, a new transaction with a freshly drawn execution time enters the system.

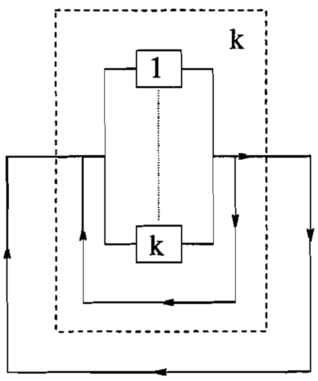


Figure 2. Queueing model of the closed system

3.1. Analysis

The closed system was analyzed by Morris and Wong [6]. They derived an approximation for the mean response time of a transaction. The analysis of Morris and Wong can be extended to an approximation for the entire distribution of the response time. In order to show this, we first restate the throughput-analysis of [6], using our own notation. Morris and Wong make the following assumption.

Commit Assumption: In the closed system with multiprogramming level k, a transaction T in execution observes other transactions to commit (leave the system) according to a Poisson process with rate α_k.

In order to determine the rate α_k at which T observes the other $(k-1)$ transactions to commit, we note that α_k is $(k-1)$ times the rate at which one CPU commits transactions. The total time a transaction spends in the system is ix if it requires i runs and has execution time x. Define $R_k(x)$ as the number of runs needed for a transaction with execution time x. Denote the response time of a transaction with execution time x by $S_k(x)$. Then $S_k(x) = R_k(x)x$. By the commit assumption, the probability that a transaction with execution time x has to be rerun is $1 - e^{-\alpha_k bx}$. Thus

$$P(R_k(x) = i) = \left(1 - e^{-\alpha_k bx}\right)^{i-1} e^{-\alpha_k bx}.$$

Hence, the expected response time of a transaction with execution time x is

$$E[S_k(x)] = \sum_{i=1}^{\infty} ixP(R_k(x) = i) = xe^{\alpha_k bx}.$$

Define S_k as the response time of a transaction in the closed system with population k. Conditioning on the exponential execution time gives $E[S_k] = \mu/(\mu - \alpha_k b)^2$. The long-term rate at which a CPU commits transactions is $1/E[S_k]$, so

$$\alpha_k = (k-1)\frac{(\mu - \alpha_k b)^2}{\mu}. \tag{1}$$

Now α_k can be solved from the quadratic equation (1). This yields

$$\alpha_k = \frac{\mu(1 + 2(k-1)b) - \mu\sqrt{1 + 4(k-1)b}}{2(k-1)b^2}, \tag{2}$$

the second root of (1) being excluded by the requirement $\mu - \alpha_k b > 0$, which is needed to ensure that $E[S_k]$ is finite. An interpretation of the requirement $\mu - \alpha_k b > 0$ is the following. Consider an arbitrary transaction T and define a T-invalidation epoch to be an epoch at which a transaction commits that conflicts with T. Then $\mu - \alpha_k b > 0$ says, that the average time needed for one run of T should not exceed the average time between two T-invalidation epochs: $1/\mu < 1/\alpha_k b$.

Next, we extend the analysis of Morris and Wong to an approximation for the distribution function of the response time S_k. Consider a transaction T with execution time x. Then

$$\begin{aligned} P(S_k(x) \le t) &= \sum_{i=1}^{\infty} P(S_k(x) \le t \mid R_k(x) = i)P(R_k(x) = i) \\ &= 1 - \left(1 - e^{-\alpha_k bx}\right)^{\lfloor \frac{t}{x} \rfloor}, \qquad x \le t. \end{aligned}$$

Hence, under the commit assumption, with α_k given by (2),

$$P(S_k \le t) = 1 - e^{-\mu t} - \int_0^t (1 - e^{-\alpha_k bx})^{\lfloor \frac{t}{x} \rfloor} \mu e^{-\mu x} \mathrm{d}x. \tag{3}$$

Two remarks must be made. First, for high values of the conflict probability b (say $b > 0.6$), it can happen that the approximation (2) for α_k returns a value smaller than $\frac{k-1}{k}\mu$. This would imply that the throughput of the system, estimated as $\frac{k}{k-1}\alpha_k$, is smaller than μ which can never happen. The closed system always contains at least one successful transaction so has a throughput of at least μ. Thus, the approximation for α_k must be corrected. It can be shown that this correction is needed when $b > (k - \sqrt{k})/(k-1)$. Morris and Wong modify (2) to read

$$\alpha_k = \begin{cases} \dfrac{\mu(1 + 2(k-1)b) - \mu\sqrt{1 + 4(k-1)b}}{2(k-1)b^2} & \text{if} \quad b \le \dfrac{k - \sqrt{k}}{k-1} \\ \dfrac{k-1}{k}\mu & \text{otherwise.} \end{cases}$$

Note that whenever this correction is necessary, OCC is not an attractive concurrency control algorithm anyway. OCC was designed for databases with a small conflict probability (see [4]).

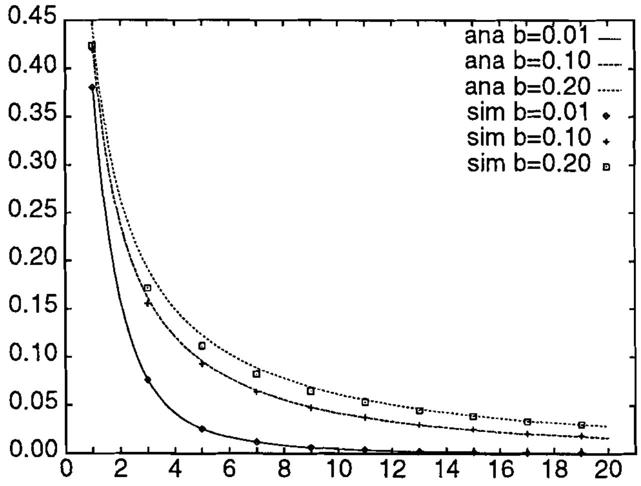
As a second remark, we note that it can be seen from (2), that α_k approaches μ/b as k approaches infinity. Also, α_k is strictly increasing in k. Hence, according to the analysis, the throughput $\frac{k}{k-1}\alpha_k$ of the closed system increases monotonically to μ/b. Thus if the number of CPUs becomes infinitely large, boundedness of the throughput implies unbounded response times.

3.2. Verifying the commit assumption

The quality of the approximation for the distribution of the response time depends on the quality of the commit assumption. Therefore, we studied the actual departure process of the closed system by simulation. According to the commit assumption, the interdeparture time (i.e., the time that elapses between two consecutive commits) is exponentially distributed with parameter $\frac{k}{k-1}\alpha_k$. Simulation experiments indicated that the expected interdeparture time is indeed approximately equal to $(k-1)/(k\alpha_k)$. Moreover, the coefficient of variation of the interdeparture time in the simulations ranges from 0.98 to 1.03 so is very close to 1. Finally, simulation showed that subsequent interdeparture times are practically uncorrelated and independent. Thus, the commit assumption seems quite realistic.

3.3. Numerical results for the response-time distribution

Since the commit assumption seems to be reasonable, we may expect that the approximation (3) of the response-time distribution is quite good. We are interested in the probability $P(S_k > t)$ that a transaction does not meet its deadline t. For good system performance, this probability must be small. In this section, we test the quality of our approximation for $P(S_k > t)$. The value of $P(S_k > t)$ produced by the analysis was compared with the simulation result, for various t and k.



(a) $P(S_{10} > t)$

	k	μ_k	
		Sim	Ana
$b = 0.01$	2	1.96	1.96
	6	5.47	5.47
	10	8.51	8.52
$b = 0.10$	2	1.67	1.68
	6	3.18	3.22
	10	3.99	4.04
$b = 0.20$	2	1.44	1.46
	6	2.24	2.29
	10	2.63	2.68

(b) μ_k

Figure 3. Simulation versus analysis of $P(S_{10} > t)$ and μ_k in closed system

For t ranging from 1 to 20, Figure 3(a) plots $P(S_{10} > t)$. The lines show the analysis results, the symbols simulation results. The different types of lines and symbols represent different values of the conflict probability: $b = 0.01$, $b = 0.1$, and $b = 0.2$. In a database of size $d = 1000$, $b = 0.01$ corresponds approximately to a transaction size $a = 3$, $b = 0.1$ to $a = 10$, and $b = 0.2$ to $a = 15$. The mean execution time $(1/\mu)$ was taken equal to 1. The numerical integration method used for evaluating (3) was adaptive Simpson's quadrature (with an error smaller than 1E-5). The figure shows that our analysis of the response-time distribution closely matches the simulation. This is also true for other values of k (< 10). The largest absolute difference found is 0.02. Figure 3(b) contains simulation and analysis results for the throughput of the closed system, denoted by μ_k. The differences between analysis and simulation are very small.

4. RESPONSE-TIME DISTRIBUTION IN THE OPEN SYSTEM

We now analyze the open system of Figure 1. Transactions arrive at the system according to a Poisson process with rate λ and transactions that find all CPUs occupied wait in a queue. The response time consists of waiting time plus the time spent executing inside the dashed area.

4.1. Analysis

The system of Figure 1 resembles an $M/G/N$ queueing system, with the exception that the service times are neither independent, nor identically distributed. As seen in Section 3, the distribution of the response time of a transaction in the dashed area depends on the number of transactions concurrently in execution. In the open system, the number of transactions concurrently in execution changes through time. Thus, the distribution of the service time in the open system (i.e., the time a transaction spends in the dashed area) is not known beforehand. As no exact expression exists for the response-time distribution of an $M/G/N$ system, there is not much hope for an exact expression for the response-time distribution in the even more complicated model of Figure 1. Therefore, an approximation for the response-time distribution is desired. Define the throughput of a closed system with k customers by μ_k ($= \frac{k}{k-1}\alpha_k$ from Section 3). The approximation we suggest consists of 3 parts:

1. Approximate the distribution of the waiting time W_i of a transaction that has i transactions waiting ahead of it by an Erlang$(i+1, \mu_N)$-distribution.
2. For $1 \leq k \leq N$, approximate the service-time distribution of a transaction that finds $k-1$ CPUs busy on entering the database by $P(S_k \leq t)$, that is, the response time of a transaction in a closed system with population k. Approximate the service-time distribution of a transaction that finds all CPUs busy on its arrival by $P(S_N \leq t)$.
3. Approximate the steady-state probability of having a total of i transactions in the system by

$$\pi_i = \begin{cases} \displaystyle\prod_{k=1}^{i} \left(\frac{\lambda}{\mu_k}\right) \pi_0 & , 1 \leq i \leq N \\ \left(\dfrac{\lambda}{\mu_N}\right)^{i-N} \pi_N & , i > N, \end{cases} \tag{4}$$

where π_0 is computed from the normalization condition $\sum_{i=0}^{\infty} \pi_i = 1$.

Support for part 1 is the fact that, as long as there are transactions waiting, the dashed area of Figure 1 behaves like a closed system with N transactions, for which the commit assumption gives a good approximation of the departure process. According to the commit assumption, the departure process is a Poisson process. Thus the time until the next departure is approximately exponentially distributed with parameter μ_N. Part 2 comes down to just ignoring that the number of concurrently executing transactions changes through time. Whether this yields a good approximation for the distribution of the time a transaction spends in the dashed area, cannot be said beforehand. Approximation part 3 is (like 1) also based on the commit assumption. Under the commit assumption, the system behaves as a state-dependent exponential server with rate μ_k when k transactions are in service. The steady-state probabilities are thus approximated by the steady-state distribution of a birth-death process with birth rate λ and death rate μ_k (with $\mu_k = \mu_N$ for $k > N$).

Using 1, 2, 3, and the PASTA [9] property, we get as approximation for the distribution of the response time S in the open system:

$$P(S > t) = \sum_{i=0}^{N-1} \pi_i P(S_{i+1} > t) + \sum_{i=N}^{\infty} \pi_i P(W_{i-N} + S_N > t), \tag{5}$$

with W_i an Erlang$(i + 1, \mu_N)$-distributed variable, and π_i given by (4).

Evaluating the second term of (5) by conditioning on W_{i-N} and by assuming independence of W_{i-N} and S_N gives

$$\sum_{i=N}^{\infty} \pi_i P(W_{i-N} + S_N > t) = \sum_{i=N}^{\infty} \pi_N \left(\frac{\lambda}{\mu_N}\right)^{i-N} \int_0^{\infty} \frac{(\mu_N y)^{i-N}}{(i-N)!} \mu_N e^{-\mu_N y} P(S_N > t - y) \mathrm{d}y.$$

Substituting the distribution of S_N yields after some algebra

$$\begin{aligned}\sum_{i=N}^{\infty} \pi_i P(W_{i-N} + S_N > t) &= \frac{\pi_N \mu_N}{\mu_N - \lambda} e^{-\mu_N t} e^{\lambda t} + \frac{\pi_N \mu_N}{\mu_N - \mu - \lambda} \left[e^{-\mu t} - e^{-(\mu_N - \lambda)t}\right] \\ &+ \pi_N \mu_N \int_0^t e^{-(\mu_N - \lambda)y} \int_0^{t-y} (1 - e^{-\alpha_N b x})^{\lfloor \frac{t-y}{x} \rfloor} \mu e^{-\mu x} \mathrm{d}x \, \mathrm{d}y. \end{aligned} \tag{6}$$

The latter double integral can be reduced to a single integral. Our approximation for the response-time distribution thus is

$$\begin{aligned} P(S > t) = {} & \sum_{i=0}^{N-1} \prod_{k=1}^{i} \left(\frac{\lambda}{\mu_k}\right) \pi_0 \left\{ \int_0^t (1 - e^{-\alpha_{i+1} b x})^{\lfloor \frac{t}{x} \rfloor} \mu e^{-\mu x} \mathrm{d}x + e^{-\mu t} \right\} + \\ + {} & \pi_N \mu_N \left\{ \frac{e^{-(\mu_N - \lambda)t}}{\mu_N - \lambda} + \frac{e^{-\mu t} - e^{-(\mu_N - \lambda)t}}{\mu_N - \mu - \lambda} \right\} + \\ + {} & \frac{\pi_N \mu_N}{\mu_N - \lambda} \int_0^t \left\{ \frac{1 - [e^{(\mu_N - \lambda)x}(1 - e^{-\alpha_N b x})]^{\lfloor \frac{t}{x} \rfloor - 1}}{1 - e^{(\mu_N - \lambda)x}(1 - e^{-\alpha_N b x})} e^{-(\mu_N - \lambda)(t - x)} (e^{(\mu_N - \lambda)x} - 1)(1 - e^{-\alpha_N b x}) \right. \\ & \left. + (1 - e^{-(\mu_N - \lambda)(t - \lfloor \frac{t}{x} \rfloor x)})(1 - e^{-\alpha_N b x})^{\lfloor \frac{t}{x} \rfloor} \right\} \mu e^{-\mu x} \mathrm{d}x. \end{aligned}$$

The remaining integrals in the expression have to be evaluated numerically. They are of the same type as the integral in the approximating expression for $P(S_k > t)$ in Section 3.

4.2. Numerical results for the response-time distribution

We tested the quality of the approximation for $P(S > t)$ against simulation, for the three cases $b = 0.01$, $b = 0.1$ and $b = 0.2$ ($\mu = 1$). In order to prevent that the number of transactions in the system explodes, the transaction arrival rate λ_N for a system with N CPUs should not exceed μ_N, the maximum throughput when all N CPUs are used. Therefore, define $\rho = \lambda_N / \mu_N$ as a measure for the load (utilization) of the system. We considered systems with $\rho = 0.5$, 0.7, 0.8, and 0.9, for which numerical results are given in Table 2(a), (b), (c), and (d), respectively. Arrival rate λ_N was computed from $\lambda_N = \rho \mu_N$. The value μ_N was obtained from the analysis of the closed system. As seen in Section 3, the analytical approximation for μ_N seems to be quite good. However for large ρ (0.9), the errors relative to $1 - \rho$ become important.

The tables contain simulation and analysis results for $E[S]$, $P(S > 2)$, $P(S > 10)$, and either $P(S > 5)$ or $P(S > 20)$. As the mean work requirement of a transaction is 1, $P(S > t)$ corresponds to the probability that a transaction spends more than t times its average required execution time in the system. Note that we evaluated the integrals in the approximation for $P(S > t)$ by adaptive Simpson's quadrature, with an error smaller than 1E-5.

The accuracy of the approximation compared to simulation is summarized in Table 1. When $b = 0.01$, the approximation is excellent for all considered values of ρ. When $b = 0.1$, the approximation is good for ρ up to 0.8, but is not accurate for $\rho = 0.9$. Finally, when $b = 0.2$ the approximation is only satisfactorily accurate for $\rho = 0.5$ and 0.7. For $b = 0.2$ and $\rho = 0.8$ or 0.9, the approximation is quite bad: the analysis differs too much from the simulation and severely underestimates $E[S]$ and $P(S > t)$. The reason for the discrepancy between analysis and simulation in these cases is twofold.

Table 1
Accuracy of the analysis compared to simulation

	b		
ρ	0.01	0.10	0.20
0.5	++	++	+
0.7	++	+	+
0.8	++	+	−
0.9	+	−	−−

The first part of the discrepancy is caused by the error we make in estimating the throughput μ_k of the closed system ($k \leq N$). In Figure 3(b) of Section 3, we saw that this error is quite small, but in open systems with a high load, a small error in the throughput estimate can result in a large error in the system performance. Since we overestimated μ_N and chose $\lambda_N = 0.9\mu_N$, the actual utilization of the open system is not 0.9 but larger. E.g. for $N = 10$: $\lambda_{10} = 0.9 \cdot 4.04 = 3.64$, and $\frac{3.64}{3.99} = 0.91$ for $b = 0.1$. For $b = 0.2$, $\lambda_{10} = 0.9 \cdot 2.68 = 2.41$, and $\frac{2.41}{2.63} = 0.92$. So the systems we looked at with presumed utilization 0.9 in reality have higher loads. As the performance measures involve the term $1 - \rho$, an error of 2 percent in ρ leads for $\rho = 0.8$ to an error in the response time S in the order of 10 percent, for $\rho = 0.9$ even to an error of 20 percent.

Even if the throughput is estimated perfectly, so if the analysis is done with values for μ_k resulting from a simulation of the closed system, there still remains a gap between analysis and simulation. This remaining second part of the discrepancy between analysis and simulation is caused by the 3-part approximation we used (as described in Section 4.1). Roughly said, this 3-part approximation accounts for two-third of the gap between analysis and simulation, and the throughput error accounts for one-third of the gap.

We investigated which of the 3 approximation parts is responsible for the bad results. It turned out from simulation experiments, that the waiting-time distribution of a transaction that has i transactions waiting ahead of it is well approximated by an Erlang$(i+1, \mu_N)$-distribution. Also, simulation showed that $P(S_k \leq t)$ $[P(S_N \leq t)]$ is a fairly good approximation for the service-time distribution of a customer that finds $k - 1$ $[N]$ CPUs busy upon arrival ($k \leq N$). So part 1 and 2 of the approximation describe the system well. Hence, the most severe error must have been made in approximation part 3: the approximation for the probability π_i that i transactions are in the system. This conclusion was confirmed when we repeated all analysis experiments without part 3 as approximation for the probabilities π_i but with simulation results for π_i (for $i \leq L$ with L large). The analysis then produced values for $P(S > t)$ that were almost identical to the simulation results for $P(S > t)$ (even though the μ_k's used were not exact).

Thus, for high system loads the approximation for $P(S > t)$ should be improved by improving the approximation for the steady-state probabilities of the model. This however is not easily done. The system resembles an $M/G/N$ system but has dependent service times of which the distribution is not known beforehand: the distribution depends on the number of transactions executing concurrently. Treating the system as an $M/G/N$ system with the service time of every customer distributed as the random variable S_N leads to a clear overestimation of the response times, so is not feasible. Furthermore, simulation experiments showed that the squared coefficient of variation c^2 of the time a transaction spends in the dashed area of Figure 1 is very large (e.g. for $\rho = 0.9$, when $b = 0.1$: $c^2 \approx 2$ for $N = 2$ up to $c^2 \approx 25$ for $N = 10$; when $b = 0.2$: $c^2 \approx 2$ for $N = 2$ up to $c^2 > 100$ for $N = 10$). Consequently, existing $M/G/N$ approximations, which are all only appropriate for $c^2 \leq 2$, cannot be used and failed when we tried them. Finally, using the first two moments of the service time from simulation, fitting a hyperexponential distribution to these moments, and modeling the open system as an $M/H_2/N$ system (for which the probabilities π_i can be determined exactly) also did not give us a good approximation for π_i.

The conclusion from the numerical experiments is, that we have a good approximation for the response-time distribution of a transaction in open systems with a load up to 0.8. For $\rho = 0.9$ and $b = 0.01$, the approximation also serves well. For $\rho = 0.8$ and $b = 0.2$, or $\rho = 0.9$ and $b = 0.1$ or $b = 0.2$, the system is so volatile that it is very difficult to find a good approximation for the response-time distribution. However, designers of real-time databases would then probably choose another (non-optimistic) concurrency control algorithm anyway, since OCC performs badly in these cases (the response-time distribution has a very fat tail).

5. CONCLUDING REMARKS

We analyzed the response-time distribution in a real-time shared-memory database with optimistic concurrency control. The analysis went beyond previous performance studies because not the average but the entire distribution of the transaction response time was considered. Knowledge about the response-time distribution is essential for the design of real-time database systems. For a given database design, the response-time distribution indicates which percentage of transactions will miss its deadline. The number of CPUs needed to guarantee that this percentage does not exceed some prespecified value can thus be obtained from an approximation for the response-time distribution.

Approximations were derived for the response-time distribution in a database with a fixed multiprogramming level (the closed system), and for the response-time distribution in a database where transactions arrive according to a Poisson process (the open system). The approximation for the response-time distribution in the closed system was within a few percent of a simulation of the closed system. For the open system with utilizations up to 80%, the approximation also performed satisfactorily compared to simulation. However for higher loads, the resulting approximation for the response time in the open system is too optimistic, so cannot be used to give performance guarantees. One reason for the bad results is that under high loads a small error in the approximation of the throughput leads to a large error in the approximation of the response-time distribution. The other reason for the large error in the approximation is the use of the flow-equivalent-server concept to approximate the steady state of the system. It is of utmost importance that an alternative to the flow-equivalent-server approximation is sought that does perform well under high system loads. Such an alternative method would be a very significant step forward in the analysis of queueing networks.

For the cases where the analysis works well, it is worthwhile to try some extensions of the model. Presently we are considering non-exponential work requirements and the so-called broadcast-OCC scheduler. Under broadcast OCC, a transaction is aborted and rerun immediately when a conflicting transaction commits.

Finally, we note that for the cases where the analysis did not do well (that is, when both the conflict probability and the system load are high), OCC is not advisable anyway, since the percentage of transactions that miss their deadline is high. Then pessimistic concurrency control schemes in which data-items are locked before they are accessed and in which only one transaction can use a data-item at a time will perform better. We intend to make an analytical comparison of the response-time distribution in databases with locking (see [2]) and OCC.

ACKNOWLEDGMENTS

We thank Maarten Bodlaender, Ad Aerts and Peter van der Stok for useful discussions and their suggestions on the manuscript. In particular, we thank Rudesindo Núñez Queija for his help on simplifying the double integral (6).

REFERENCES

1. R. Agrawal, M.J. Carey, and M. Livny. Concurrency control performance modeling: alternatives and implications, ACM Transactions on Database Systems 12 (1987), pp. 609–654.
2. M.P. Bodlaender, S.A.E. Sassen, P.D.V. van der Stok, and J. van der Wal. The response time distribution in a multi-processor database with single queue static locking, Proceedings Fourth Int. Workshop on Parallel and Distributed Real-Time Systems, April 1996, Honolulu, Hawaii, pp. 118–121.
3. L. Kleinrock and F. Mehović. Poisson winner queues, Performance Evaluation 14 (1992), pp. 79–101.
4. H. Kung and J. Robinson. On optimistic methods for concurrency control, ACM Transactions on Database Systems 6 (1981), pp. 213–226.
5. Menascé, D.A. and T. Nakanishi. Optimistic versus pessimistic concurrency control mechanisms in database management systems, Information Systems 7 (1982), pp. 13–27.
6. R.J.T. Morris and W.S. Wong. Performance analysis of locking and OCC algorithms, Performance Evaluation 5 (1985), pp. 105–118.
7. C. Papadimitriou. The Theory of Database Concurrency Control, Computer Science Press, Rockville, Md., 1986.
8. Y.C. Tay. Locking Performance in Centralized Databases, Academic Press, Inc., Orlando, Florida, 1987.
9. R.W. Wolff. Poisson arrivals see time averages, Operations Research 30 (1982), pp. 223–231.
10. P.S. Yu, D.M. Dias, and S.S. Lavenberg. On the analytical modeling of database concurrency control, Journal of the ACM 40 (1993), pp. 831–872.

Table 2: Response-time distribution in open system

				$E[S]$		$P(S > 2)$		$P(S > 5)$		$P(S > 10)$	
	N	λ_N	μ_N	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana
$b = 0.01$	2	0.98	1.96	1.35	1.35	0.23	0.23	0.02	0.02	0.00	0.00
	6	2.73	5.47	1.08	1.09	0.15	0.15	0.01	0.01	0.00	0.00
	10	4.26	8.52	1.09	1.10	0.15	0.15	0.02	0.02	0.00	0.00
$b = 0.10$	2	0.84	1.68	1.50	1.49	0.25	0.25	0.04	0.04	0.01	0.01
	6	1.61	3.22	1.43	1.43	0.19	0.19	0.05	0.04	0.01	0.01
	10	2.02	4.04	1.58	1.55	0.20	0.19	0.06	0.05	0.02	0.02
$b = 0.20$	2	0.73	1.46	1.64	1.63	0.26	0.26	0.05	0.05	0.01	0.01
	6	1.15	2.29	1.71	1.66	0.21	0.20	0.06	0.05	0.02	0.02
	10	1.34	2.68	1.90	1.81	0.22	0.20	0.07	0.06	0.03	0.02

(a) $\rho = 0.5$

				$E[S]$		$P(S > 2)$		$P(S > 5)$		$P(S > 10)$	
	N	λ_N	μ_N	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana
$b = 0.01$	2	1.37	1.96	2.01	1.99	0.38	0.38	0.08	0.07	0.01	0.00
	6	3.83	5.47	1.27	1.27	0.19	0.20	0.02	0.02	0.00	0.00
	10	5.97	8.52	1.20	1.20	0.17	0.17	0.02	0.02	0.00	0.00
$b = 0.10$	2	1.17	1.68	2.42	2.26	0.41	0.41	0.13	0.11	0.02	0.01
	6	2.25	3.22	1.85	1.85	0.25	0.26	0.07	0.07	0.02	0.02
	10	2.83	4.04	2.00	1.99	0.23	0.23	0.08	0.07	0.03	0.03
$b = 0.20$	2	1.02	1.46	2.84	2.52	0.43	0.43	0.16	0.14	0.04	0.03
	6	1.60	2.29	2.36	2.31	0.27	0.28	0.10	0.09	0.04	0.03
	10	1.88	2.68	2.66	2.54	0.25	0.25	0.10	0.09	0.05	0.04

(b) $\rho = 0.7$

				$E[S]$		$P(S > 2)$		$P(S > 10)$		$P(S > 20)$	
	N	λ_N	μ_N	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana
$b = 0.01$	2	1.57	1.96	2.88	2.83	0.51	0.51	0.03	0.02	0.00	0.00
	6	4.37	5.47	1.56	1.55	0.27	0.27	0.00	0.00	0.00	0.00
	10	6.82	8.52	1.37	1.37	0.21	0.21	0.00	0.00	0.00	0.00
$b = 0.10$	2	1.34	1.68	3.65	3.24	0.55	0.54	0.07	0.05	0.01	0.00
	6	2.57	3.22	2.50	2.36	0.35	0.36	0.04	0.03	0.01	0.01
	10	3.24	4.04	2.44	2.43	0.28	0.30	0.04	0.04	0.01	0.01
$b = 0.20$	2	1.17	1.46	4.68	3.66	0.58	0.56	0.12	0.07	0.03	0.01
	6	1.83	2.29	3.39	3.06	0.38	0.39	0.06	0.05	0.02	0.02
	10	2.14	2.68	3.43	3.27	0.31	0.33	0.06	0.05	0.02	0.02

(c) $\rho = 0.8$

				$E[S]$		$P(S > 2)$		$P(S > 10)$		$P(S > 20)$	
	N	λ_N	μ_N	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana
$b = 0.01$	2	1.76	1.96	5.62	5.36	0.71	0.71	0.16	0.15	0.03	0.02
	6	4.92	5.47	2.52	2.43	0.47	0.47	0.02	0.01	0.00	0.00
	10	7.67	8.52	2.01	1.92	0.37	0.36	0.01	0.01	0.00	0.00
$b = 0.10$	2	1.51	1.68	7.61	6.21	0.74	0.73	0.26	0.20	0.08	0.04
	6	2.89	3.22	5.30	3.91	0.59	0.56	0.15	0.08	0.04	0.01
	10	3.64	4.04	4.74	3.68	0.51	0.49	0.12	0.06	0.04	0.02
$b = 0.20$	2	1.31	1.46	11.6	7.08	0.78	0.74	0.38	0.25	0.18	0.06
	6	2.06	2.29	13.4	5.25	0.66	0.60	0.29	0.14	0.15	0.03
	10	2.41	2.68	8.12	5.22	0.56	0.52	0.20	0.11	0.10	0.03

(d) $\rho = 0.9$