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

Eindhoven University
of Technology

Department of Mathematics
and Computing Sciences

Memorandum COSOR 97-22

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

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

Eindhoven, December 1997
The Netherlands

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

Abstract

For a real-time shared-memory database with optimistic concurrency control, an approximation for the distribution of the transaction response time and thus for the deadline miss probability is obtained. Transactions arrive at the database according to a Poisson process. There is a limited number of CPUs that can handle transactions in parallel. Transactions have soft deadlines, and the probability of data conflicts is equal for all transactions. No restrictive assumptions are needed about the execution time of transactions: it can have any general probability distribution. We approximate the behavior of the system by a multi-server queue with a special type of feedback. The analysis of this queueing system is based on an interpolation of the corresponding systems with constant and exponential execution times. Numerical experiments, which compare the approximative analysis with a simulation of the database, show that the approximation of the response-time distribution is quite accurate and thus very useful for real-time database design.

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 maximum throughput of the database should be as large as possible. 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 temporarily overloaded, but at least a certain fraction of the transactions should meet some prescribed deadline. In a real-time database (RTDB), both consistency and timeliness are important. In this paper, we investigate soft real-time databases and are interested in an *analytical* method for computing the probability that a transaction meets its deadline. (For a *simulation* study of this probability, see Chapter 16 in [6].)

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 can conflict if they access the same data-item, at least one of them with the intention to write. To execute conflicting transactions, a concurrency control scheme is needed. Concurrency control schemes govern the simultaneous execution of transactions such that overall correctness of the database is maintained (see e.g. [11]). 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 high transaction response times.

When the conflict probability is low, it can be advantageous to use the optimistic concurrency control (OCC) scheme proposed by KUNG and ROBINSON [7] . Under OCC, all CPUs can be used for

transaction processing at the same time, even for processing conflicting transactions. 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. During the execution phase, all actions T performs on data-items are only done on local copies of the data-items. 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 (that is, if at least one of the data-items read by T was in the meantime changed globally by another transaction), T must be rerun. The local changes T made to data-items then don't become global but are erased. If no conflicts occurred, T completes the validation phase successfully and enters the commit phase, where the data-items used by T are updated globally.

Despite the existence of extensive simulation studies of the performance of OCC compared to locking, with clear recommendations as to in what situations OCC performs better than locking and vice versa, OCC is still not accepted in practice. In RTDBs with soft deadlines, OCC is preferable to locking if resources (CPUs) are not the limiting factor ([1]). For RTDBs with firm deadlines, where transactions that miss their deadline are discarded immediately, HARITSA, CAREY, and LIVNY [3] concluded that OCC generally performs much better than locking, even if resources are not plentiful.

Hence there seems to be no good reason why OCC is not accepted in practice. This is a motivation for us to come up with an *analysis* of OCC for RTDBs, as opposed to all existing and time-consuming *simulation* studies. We start by analyzing soft deadlines. In later work we will analyze OCC systems with firm deadlines.

So we are interested in an analytical method to evaluate the performance of a real-time database with OCC. The method should provide an accurate estimate of the percentage of transactions that meet their deadlines. Existing analytical performance studies for OCC ([9], [10], [5], and [16]) only consider *average* system performance, such as throughput, average response time, and the average number of restarts needed for a transaction. (The response time of a transaction is the total time between its arrival and its commit.) Knowledge about average response times is not enough to estimate the probability that a transaction meets its deadline: for this, an approximation of the response-time *distribution* is required. As far as we know, no analytical performance studies of real-time databases with OCC exist that address the distribution of the response time.

In [13], we analyzed a RTDB with OCC in which the execution times of transactions are *exponentially* distributed. A follow-up on that study was the paper [14], in which we analyzed a RTDB with *constant* execution times. The present paper tackles the *general* case, where the execution time of a transaction can have any probability distribution. The general case permits no simple analytical direct solution, not even for the average response time. Therefore, the analysis of the general case is done by an interpolation between the systems with constant and exponential execution times. The idea for such a system-interpolation originates from existing approximation methods in queueing theory. Response times in analytically intractable queueing models with general service times have been ap-

proximated successfully by interpolation, using the response times of simpler and exactly analyzable queueing models with deterministic and exponential service times as building blocks. An excellent overview of system-interpolations for multi-server queues is given by KIMURA [4].

The paper is organized as follows. The model for a RTDB with OCC and general execution times is explained in Section 2. Section 3 briefly describes the analyses of the special cases with constant and exponential execution times, respectively. In order to develop some intuition about the behavior of the system with general execution times compared to the systems with exponential and constant execution times, we perform a simulation study in Section 4. Based on this intuition, in Section 5 we present the approximative analysis of the general case. Numerical results, which compare analysis with simulation, are given in Section 6. Finally, Section 7 contains some concluding remarks.

2 The Model

In the introduction, we described the OCC scheme in detail. In this section, 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.

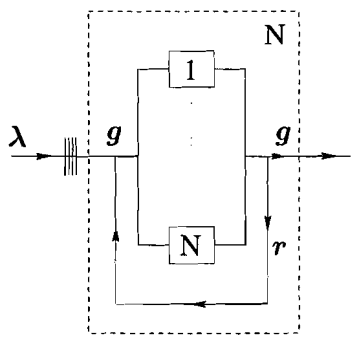


Figure 1: *Queueing model of the system*

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). It is assumed that the commit phase takes negligible time compared to execution plus validation, and that validation can be efficiently done in parallel. The assumption that commit takes negligible time compared to execution plus validation is reasonable, since we consider a system where all data-items are in main memory so where no disks are attached.

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 first in queue is taken into execution. We also refer to execution plus validation as one *transaction run.*

The time needed for one transaction run is called the *execution time.* The execution time can have any general probability distribution. For our analysis of the system it is sufficient to have an estimate

of the mean and the standard deviation of the execution time. We denote the average execution time by $1/\mu$.

With regard to transaction behavior, we assume that two transactions conflict with probability b. The conflict probability b is an input parameter for characterizing the amount of data contention in the system. The value of b is larger when the (number of data-items in the) database is smaller or when transactions access more data-items. It is possible to extend the model and the analysis to handle non-uniform data access, but this will be addressed in a forthcoming paper.

The queueing model of Figure 1 is no standard feedback model: it is unconventional in three respects. Firstly, a rerun of a transaction requires *exactly* the same amount of time as the first run. Secondly, when a transaction is fed back, it does not have to rejoin the queue to await a new service but it is restarted immediately. Thirdly, 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. Since it is an open system, the number of concurrently executing transactions varies during T's execution.

For an exact analysis of the queueing model, it is convenient to label the transactions in service by colors, say green and red. A transaction T is green at the start of every run. During its run, T is colored red as soon as a transaction commits that conflicts with T. A red transaction discovers at its validation that it has to be rerun (it then returns to the CPU as a green transaction); a transaction that is still green at validation time is allowed to commit. In this way, the color of a transaction at validation time determines whether the transaction must be rerun. The colors red and green are depicted in Figure 1 as r and g, respectively.

Using this colorful representation of transactions, the state of the queueing model at time t is *exactly* described by the number $w(t)$ of waiting transactions at time t, and for CPU i ($i = 1, \ldots, N$) the color $c_i(t)$ (red, green), the remaining execution time $r_i(t)$ and the execution time $x_i(t)$ of the transaction at CPU i at time t. So by the vector $(w(t), c_1(t), r_1(t), x_1(t), \ldots, c_N(t), r_N(t), x_N(t))$. With this state-description, a *simulation* program of the queueing model is easily made. However, an exact *analysis* with this state-description seems so intractable that we are not very optimistic about the chances of finding one. Therefore, we will propose an approximative analysis of the queueing model.

For convenience, we introduce the following notation. For the case with generally distributed execution times, the queueing system of Figure 1 is denoted by the four-part code $M/G/N/OCC$. The special cases with deterministic and exponential execution times are denoted by $M/D/N/OCC$ and $M/M/N/OCC$, respectively. The notation corresponds with the well-known notation for queueing systems. That is, the first symbol specifies the distribution of the interarrival times of transactions (M: Markovian, exponential), the second symbol specifies the distribution of the amount of work required by a transaction (G: general), and the third symbol specifies the number of CPUs (servers). The fourth symbol indicates that the service discipline is special, namely, OCC. In the next section, we give a brief

account of the analyses of the $M/D/N/OCC$ and the $M/M/N/OCC$ system.

3 Analysis of Two Special Cases

3.1 $M/D/N/OCC$

The analysis of the response-time distribution in an $M/D/N/OCC$ system is reported in detail in [14]. Here, we only briefly give the algorithm. The analysis consists of three approximation steps.

The first step is an approximation for the feedback mechanism. Consider a transaction T that found $n-1$ other transactions in execution when it started its transaction run. Instead of registrating whether a conflicting transaction committed during the execution of T, we approximate the success probability $p(n)$ of T (i.e., the probability that T validates successfully) by looking at a closed system with a constant number of n transactions in execution.

A quick but accurate approximation for the success probability $p(n)$ in the closed system is as follows. Suppose that all transactions in the closed system independently have a success probability of $p(n)$. Then every transaction that validates during T's execution colors T red (makes T unsuccessful) with probability $bp(n)$. Since exactly $n-1$ transactions validate during T's execution, T's success probability $p(n)$ is the unique fixed point of the equation

$$p(n) = (1 - bp(n))^{n-1}$$

on the interval $(0, 1]$.

So, using the probabilities $p(n)$ as success probabilities, the $M/D/N/OCC$ system is approximated by the so-called $M/D/N$ queue with starting-state dependent feedback, shown in Figure 2.

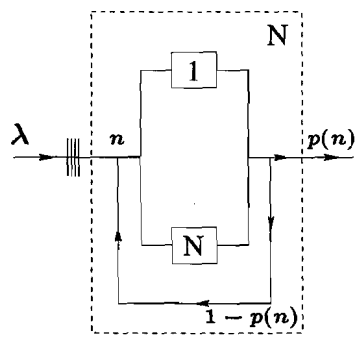


Figure 2: *M/D/N queue with starting-state dependent feedback*

The second step is an approximative analysis of the steady-state probabilities of the $M/D/N$ queue with starting-state dependent feedback. We developed two different methods to compute the steady-state probabilities of this new queueing model (see also [12]). Although both methods give satisfactorily accurate approximations for the steady state of the queue, for reasons of brevity we only mention

the second method here. The method resembles the exact analysis of CROMMELIN [2] for the $M/D/c$ queue by observing the state of the system after every D ($= 1/\mu$) time units. Define

$a[\ell]$ as the probability that ℓ transactions arrive in $(t, t+D]$, so $a[\ell] = e^{-\lambda D}(\lambda D)^{\ell}/\ell!$,

B_i^j as the probability that i transactions depart (are successful) during a time-interval $(0, D]$, given that j transactions are present at the start of the interval. We approximate B_i^j by $\binom{N}{i} p(N)^i (1-p(N))^{N-i}$ if $j \geq N$ and by $\binom{j}{i} p(j)^i (1-p(j))^{j-i}$ if $j < N$.

Then, by relating the number of transactions in the system at time t to the number at time $t+D$, the (approximate) steady-state probability φ_k of having k transactions in the system can be found from the linear equations

$$\begin{aligned} \varphi_k &= \sum_{j=0}^{N+k} \varphi_j \sum_{i=\max\{0,j-k\}}^{\min\{j,N\}} B_i^j a[k-j+i], \qquad k \geq 0, \\ \sum_{k=0}^{\infty} \varphi_k &= 1. \end{aligned} \tag{1}$$

There exists a large M such that for $k \geq M$, $\varphi_k \approx \varphi_M \tau^{k-M}$, where τ is the unique root of the equation $1 - p(N)(1-y) = \exp(\lambda D(1-1/y)/N)$ on the interval $(0,1)$. Hence, the system (1) can be solved numerically by choosing a large M and substituting $\varphi_k = \varphi_M \tau^{k-M}$ for $k \geq M$.

The third step is the approximation of the response-time distribution of the $M/D/N$ queue with starting-state dependent feedback, given the steady-state probabilities computed in step 2. Let S denote the response time of a transaction, that is, the total time a transaction spends in the system. Then, using Little's theorem ([8]), we approximate the average response time by $E[S] = \frac{1}{\lambda} \sum_k k \varphi_k$.

The response time S consists of waiting time and total service time (i.e., a number of transaction runs). We approximate the distribution of the total service time of a transaction that sees i other CPUs busy at the start of its first execution run by DG_{i+1}, with G_j a geometrically distributed random variable with success probability $p(j)$. The waiting-time distribution $P(W \leq x)$ is approximated similarly to Crommelin's exact analysis of the waiting time in the $M/D/c$ queue. The algorithm for computing $P(W \leq x)$ is given in Appendix A. The interpretation of both the notation and the algorithm can be found in [12]. For the resulting approximating formula for the response-time distribution $P(S \leq t)$, we refer to Appendix A.

Numerical experiments for various system loads and conflict probabilities indicate, that the approximation produces reasonably accurate estimates for the transaction response-time distribution, even for high system loads.

3.2 $M/M/N/OCC$

In [13], we report the analysis of the $M/M/N/OCC$ model in detail. Again, the analysis consists of three approximation steps, which we briefly describe below.

The first step is an approximation for the throughput μ_n of a closed OCC system with a constant number of n transactions in execution. In accordance with MORRIS and WONG [10], we approximate the commit process of the closed system by a Poisson process with as rate the unknown μ_n. Using the commit assumption, the success probability of an execution run of length x equals $e^{-(n-1)\mu_n bx/n}$. Thus, the mean response time $E[B_n]$ in the closed system equals $\mu/(\mu - \frac{(n-1)}{n}\mu_n b)^2$ if $b \leq \frac{n-\sqrt{n}}{n-1}$, and ∞ otherwise. Also, $\mu_n = n/E[B_n]$. Solving μ_n from these two relations yields

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

The second step is an approximative analysis of the steady-state probabilities φ_k of the $M/M/N/OCC$ system. We approximate φ_k by the steady state of the queueing model of Figure 3.

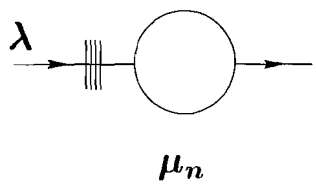


Figure 3: *Flow-equivalent server*

Since Figure 3 represents a birth-death process with birth rate λ and death rate μ_n ($n \leq N$),

$$\varphi_k = \begin{cases} \displaystyle\prod_{n=1}^{k} \left(\frac{\lambda}{\mu_n}\right) \varphi_0 & , 1 \leq k \leq N \\ \left(\dfrac{\lambda}{\mu_N}\right)^{k-N} \varphi_N & , k > N, \end{cases}$$

where φ_0 is computed from the normalization condition $\sum_{k=0}^{\infty} \varphi_k = 1$.

The third step is the approximation of the response-time distribution, using the results of step 1 and 2. The average response time is approximated by $E[S] = \frac{1}{\lambda}\sum_k k\,\varphi_k$. We approximate the distribution of the total service time of a transaction that sees i other CPUs busy at the start of its first service run by the distribution of B_{i+1}, the response time of a transaction in the closed system with population $i+1$. That is, by

$$P(B_{i+1} \leq t) = 1 - e^{-\mu t} - \int_0^t (1 - e^{-\mu_{i+1} bxi/(i+1)})^{\lfloor \frac{t}{x} \rfloor} \mu e^{-\mu x} \mathrm{d}x.$$

The waiting-time distribution of a transaction that finds $i \geq N$ transactions in the system upon arrival is approximated by an Erlang$(i+1-N, \mu_N)$-distribution. Together, this leads to an approximation for the response-time distribution. The resulting formula for $P(S \leq t)$ is given in Appendix B.

Numerical experiments for various system loads and conflict probabilities show, that the approximation is accurate for systems with a utilization ρ up to 0.8. For $\rho = 0.9$ and $b = 0.01$, the approximation also serves well. For $\rho = 0.9$ and $b = 0.1$, the system is so volatile that it is very difficult to find a good approximation for the response-time distribution. More about this in the next section.

4 Influence of the Execution-Time Distribution

In our study [13], we found that in the $M/M/N/OCC$ system with $b \geq 0.1$ some response times become extremely large when the system load is above 0.8. The reason for this is twofold.

First, the combination of a high conflict probability and a large arrival rate of transactions makes that a long transaction is almost always invalidated during its run by other (short) transactions. Since a rerun takes exactly the same large amount of time as the first run, a long transaction has almost no chance to successfully leave the system.

Second, during all of their long and unsuccessful runs, long transactions unnecessarily keep CPUs busy. Under high system loads, this loss of precious CPU capacity has large consequences for the response times of all other transactions in the system.

The $M/D/N/OCC$ system does not show this dramatically bad performance because there are no long transactions: all execution times are equal to $1/\mu$. Thus, comparing the $M/D/N/OCC$ and $M/M/N/OCC$ systems, the variance of the execution-time distribution has a large influence on the (tails of the) response-time distribution.

In this section, we try to quantify the influence the execution-time distribution has on the behavior of the $M/G/N/OCC$ system. Using existing analytic approaches from queueing theory, we develop an approximation conjecture about the influence of the execution time. A simulation study of the $M/G/N/OCC$ system is done to verify the correctness of the conjecture.

Squared Coefficient of Variation

Let the random variable X denote the execution time. The mean and variance of X are denoted by $E[X]$ and $\mathrm{Var}(X)$, respectively. Further, the squared coefficient of variation c_X^2 is defined as

$$c_X^2 = \frac{\mathrm{Var}(X)}{E^2[X]}.$$

The squared coefficient of variation is a measure of the variability of the random variable X. For example, $c_X^2 = 0$ if X is deterministic ($M/D/N/OCC$ system) and $c_X^2 = 1$ if X has an exponential distribution ($M/M/N/OCC$ system).

Relation of $M/G/N/OCC$ to $M/G/N$

We have analyzed the behavior of the systems with $c_X^2 = 0$ and $c_X^2 = 1$ (see Section 3). Now we are interested in the behavior of the $M/G/N/OCC$ system for other values of c_X^2. Since in an OCC system with $c_X^2 = 1$ the response times can already become undesirably large, we think OCC is not the appropriate concurrency control algorithm if the execution times have a squared coefficient of variation larger than 1. Hence, in the remainder we assume that the $M/G/N/OCC$ system we want to analyze has $0 \leq c_X^2 \leq 1$.

An exact analysis of the $M/G/N/OCC$ system seems impossible. For $b = 0$, so when there are no data conflicts and there is no feedback, the system reduces to the simpler $M/G/N$ queue. But even

for the $M/G/N$ queue no exact formula is known for the average response time, let alone for the distribution of the response time. However, in developing an approximative analysis for the $M/G/N/OCC$ queue, we can learn from the many approximations developed for the $M/G/N$ queue. An excellent survey of approximations for the $M/G/N$ queue, together with new insights, is given by KIMURA [4].

Waiting times and response times in the analytically intractable $M/G/N$ queue have been approximated successfully by interpolation, using the simpler and exactly analyzable $M/D/N$ and $M/M/N$ queues as building blocks. The idea for such interpolations originated from the *exact* relation

$$E[W_{M/G/1}] = (1 - c_B^2)E[W_{M/D/1}] + c_B^2 E[W_{M/M/1}] \tag{2}$$

for the $M/G/1$ queue where the service time B has a squared coefficient of variation c_B^2. That is, given λ and $E[B]$, the expected waiting time in an $M/G/1$ queue can be computed by a linear combination of the expected waiting time in an $M/D/1$ queue and an $M/M/1$ queue with the same λ and $E[B]$. The weights of the linear combination are determined by c_B^2. Note the remarkable fact that $E[W_{M/G/1}]$ depends on the service-time distribution only through $E[B]$ and c_B^2, so the actual shape of the service-time distribution doesn't matter. Moreover, in this interpolation c_B^2 need not lie between 0 and 1: the relation is exact for any c_B^2.

Applying relation (2) to multi-server queues, an *approximative* relation for the average waiting time in the $M/G/N$ queue is

$$E[W_{M/G/N}] = (1 - c_B^2)E[W_{M/D/N}] + c_B^2 E[W_{M/M/N}],$$

where every system has the same load $\rho = \lambda E[B]/N$. This approximation is quite accurate for $0 \leq c_B^2 \leq 2$ but should not be used for large c_B^2 (cf. [15]).

Moreover, even the percentiles $\xi(\alpha)$ of the waiting-time distribution of the $M/G/N$ queue have been successfully approximated by system interpolation (for $c_B^2 \leq 2$). The α-percentile $\xi(\alpha)$ of the random variable W is such that $P(W \leq \xi(\alpha)) = \alpha$. The interpolation formula for the percentiles of W is

$$\xi_{M/G/N}(\alpha) = (1 - c_B^2)\xi_{M/D/N}(\alpha) + c_B^2 \xi_{M/M/N}(\alpha).$$

(Interpolation on the percentiles of W was done because a direct interpolation on the waiting-time probabilities $P(W \leq t)$ did not give accurate results.)

Approximation for $M/G/N/OCC$

Analogously to the existing approximations for the $M/G/N$ queue, we would like to try such an interpolation to approximate the average response time in the $M/G/N/OCC$ queue (and later also the percentiles). The interpolation we would like to propose is

$$E[S_{M/G/N/OCC}] = (1 - c_B^2)E[S_{M/D/N/OCC}] + c_B^2 E[S_{M/M/N/OCC}],$$

but then we immediately encounter two problems.

The first problem is most apparent and concerns the weighting coefficients in the interpolation. Namely, what is c_B^2 in an $M/G/N/OCC$ system? The total service time B needed by a transaction consists of a number of runs which is not known beforehand. We only know $E[X]$ and c_X^2, and not the (distribution of the) total service time B. The distribution of B depends on the number of transactions in the system, so on the arrival rate λ.

The second problem concerns the so-called 'corresponding' $M/D/N/OCC$ and $M/M/N/OCC$ systems that are used as building blocks for the interpolation. What does 'corresponding' mean for an OCC system? In $M/G/N$ interpolations, 'corresponding' means that the building blocks $M/D/N$ and $M/M/N$ have the same load $\rho = \lambda E[B]/N$ as the $M/G/N$ system. As long as the input values λ and $E[B]$ are the same in the $M/G/N$, $M/D/N$, and $M/M/N$ queues, the load is the same. In the $M/G/N/OCC$, $M/D/N/OCC$, and $M/M/N/OCC$ queue, however, having the same λ, $E[X]$ and c_X^2 as input typically does *not* result in the same load ρ. The reason is that the expected total service time $E[B]$ is not the same in the OCC models. Hence, it is difficult to find the 'corresponding' $M/D/N/OCC$ and $M/M/N/OCC$ queues.

Summarizing, problems occur in designing an interpolation, because, in contrast to the ordinary $M/G/N$ queue, the (distribution of the) total service time B in an $M/G/N/OCC$ system is not part of the input (only the execution time X is), but depends on the arrival rate λ. To express this dependence on λ, in the remainder of this section we denote the total service time by $B(\lambda)$.

Let us define the load ρ of an $M/G/N/OCC$ system as

$$\rho = \frac{\lambda E[B(\lambda)]}{N}.$$

We believe that, in a system interpolation for the $M/G/N/OCC$ queue, the loads of the building blocks $M/D/N/OCC$ and $M/M/N/OCC$ should be equal to the load of the $M/G/N/OCC$ queue, just as in the $M/G/N$ interpolations. Further, to represent the influence of the execution time on the $M/G/N/OCC$ system, we propose the weighting coefficients of the system interpolation to be $1-c_X^2$ and c_X^2. Thus, using the subscripts G, D, and M to denote general, deterministic, and exponential execution times, we come up with the following approximation conjecture.

Approximation Conjecture for the $M/G/N/OCC$ System

Consider an $M/G/N/OCC$ system with arrival intensity λ_G, conflict probability b, and execution time X with mean $E[X]$ and squared coefficient of variation c_X^2, with $0 \leq c_X^2 \leq 1$. Suppose the value of the load ρ of the system is known.

Find arrival intensities λ_D and λ_M for the $M/D/N/OCC$ and $M/M/N/OCC$ system with conflict probability b and average execution time $E[X]$, such that both systems have the same load ρ as the $M/G/N/OCC$ system. So find λ_D and λ_M such that

$$\rho = \frac{\lambda_D E[B_D(\lambda_D)]}{N} = \frac{\lambda_M E[B_M(\lambda_M)]}{N} \left(= \frac{\lambda_G E[B_G(\lambda_G)]}{N} \right).$$

The behavior of the $M/G/N/OCC$ system can then be approximated as

$$E[B_G(\lambda_G)] = (1 - c_X^2)E[B_D(\lambda_D)] + c_X^2 E[B_M(\lambda_M)]$$
$$E[W_G(\lambda_G)] = (1 - c_X^2)E[W_D(\lambda_D)] + c_X^2 E[W_M(\lambda_M)]$$
$$E[S_G(\lambda_G)] = (1 - c_X^2)E[S_D(\lambda_D)] + c_X^2 E[S_M(\lambda_M)]$$

and even

$$\xi_G(\alpha) = (1 - c_X^2)\xi_D(\alpha) + c_X^2 \xi_M(\alpha)$$

for the α-percentile of the response-time distribution.

So the Approximation Conjecture says that if the load ρ of the $M/G/N/OCC$ system is known, the behavior of the system can be approximated by a linear combination of the behavior of an $M/D/N/OCC$ and $M/M/N/OCC$ system with the same load ρ. How to determine the load ρ of the $M/G/N/OCC$ queue will be discussed later. We first do a simulation study in order to find out if the Approximation Conjecture makes sense.

Simulation Study of $M/G/N/OCC$

We studied the $M/G/N/OCC$ system by simulation (using the 'colored' state description as explained in Section 2), for execution-time distributions with a squared coefficient of variation between 0 and 1. Only $E[X]$ and c_X^2 ($0 \leq c_X^2 \leq 1$) needed to be specified. The probability distribution used for generating the execution times with mean $E[X]$ and squared coefficient of variation c_X^2 was a mixture of Erlang distributions. That is, an Erlang$(k-1, \nu)$ distribution with probability r and an Erlang(k, ν) distribution with probability $1 - r$, such that

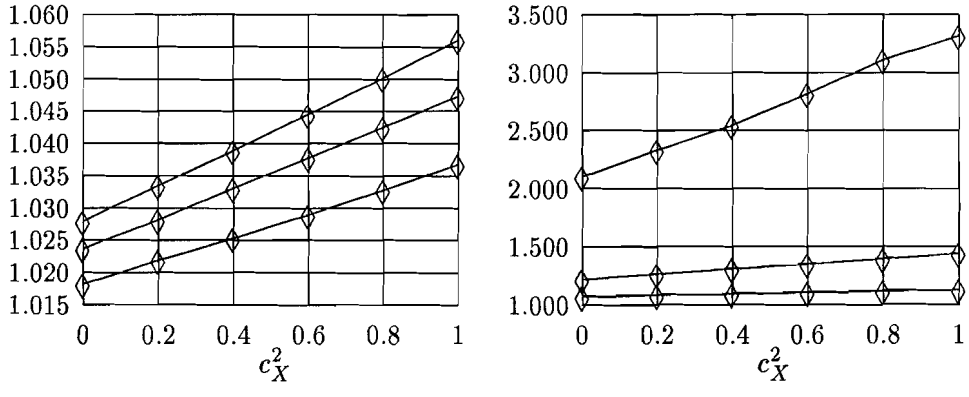
$$\frac{1}{k} \leq c_X^2 \leq \frac{1}{k-1}, \quad r = \frac{1}{1 + c_X^2}[kc_X^2 - \sqrt{k(1 + c_X^2) - k^2 c_X^2}], \quad \text{and} \quad \nu = \frac{k - r}{E[X]}.$$

Since the Approximation Conjecture relates the $M/G/N/OCC$ system to deterministic and exponential OCC systems with the same value of ρ, we wanted to simulate OCC systems with different c_X^2 but with the same ρ. We investigated 72 examples of $M/G/N/OCC$ systems. That is, we looked at $N = 4$ and $N = 10$, with $b = 0.01$ or $b = 0.1$. We varied ρ as 0.5, 0.7, and 0.9. For each value of ρ, we simulated systems with $c_X^2 = 0, 0.2, 0.4, 0.6, 0.8$, and 1. In all simulations, we took $E[X] = 1$. As there is no direct analytic relation between ρ and λ_G available and the simulation needs λ_G as input, an iterative search procedure (bisection) was used to find λ_G such that the system had the desired load ρ.

For a system with 4 CPUs (so $N = 4$) and $b = 0.01$, Table 1 shows the values of λ_G corresponding with the desired values of ρ. Figure 4 (a) and (b) show $E[B_G(\lambda_G)]$ and $E[S_G(\lambda_G)]$, respectively. The three lines with symbols in both subfigures correspond with the three values 0.5, 0.7 and 0.9 of the load ρ.

c_X^2	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$
0	1.9637	2.7339	3.5038
0.2	1.9553	2.7201	3.4840
0.4	1.9469	2.7071	3.4591
0.6	1.9386	2.6948	3.4460
0.8	1.9355	2.6817	3.4325
1.0	1.9288	2.6698	3.4058

Table 1: $N = 4$, $b = 0.01$. *Arrival rates* λ_G *for various combinations of* c_X^2 *and* ρ



(a) $E[B_G(\lambda_G)]$ (b) $E[S_G(\lambda_G)]$

Figure 4: $N = 4$, $b = 0.01$. *Simulation results for* $\rho = 0.5, 0.7, 0.9$ *as a function of* c_X^2

Both Figure 4 (a) and (b) clearly show that the behavior of the $M/G/N/OCC$ system — that is, the expected total service time $E[B_G(\lambda_G)]$ and the expected response time $E[S_G(\lambda_G)]$ — changes (almost) linearly with c_X^2. This seems to be true for all 3 investigated values of ρ. Thus for this system, with regard to average system behavior, the Approximation Conjecture appears to hold!

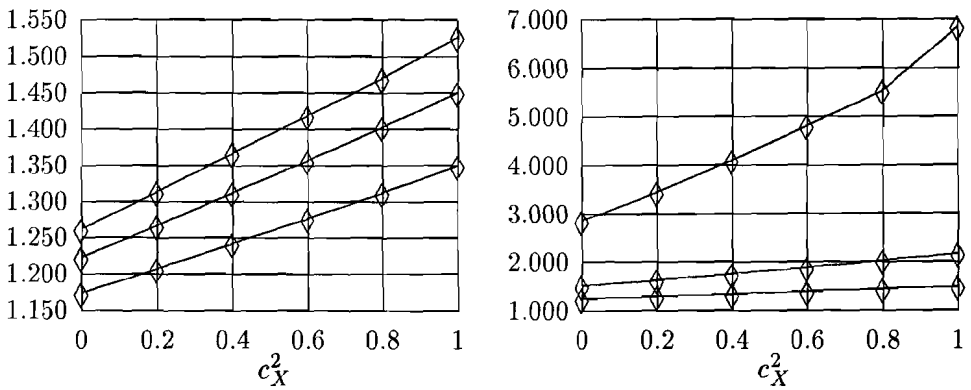
For the same system with 4 CPUs, but with $b = 0.1$, Table 2 shows the values of λ_G corresponding with $\rho = 0.5, 0.7$, and 0.9. $E[B_G(\lambda_G)]$ and $E[S_G(\lambda_G)]$ are plotted in Figure 5 (a) and (b). Again it is clear from the figures, that $E[B_G(\lambda_G)]$ is a linear function of c_X^2 for all 3 values of ρ. Hence, with regard to $E[B_G(\lambda_G)]$, the Approximation Conjecture appears to hold even for $b = 0.1$. The Approximation Conjecture is also good for $E[S_G(\lambda_G)]$: for $\rho = 0.5$ and $\rho = 0.7$, $E[S_G(\lambda_G)]$ is (almost) linear in c_X^2. However for $\rho = 0.9$ $E[S_G(\lambda_G)]$ increases more than linearly with c_X^2.

Thus, from these simulation experiments with various b, c_X^2, and ρ, we have reason to believe that the Approximation Conjecture is true for $E[B_G(\lambda_G)]$, and a good approximation for $E[S_G(\lambda_G)]$. Note that, since the waiting time is the difference between response and service time, the behavior of the not-shown $E[W_G(\lambda_G)]$ corresponds with the behavior of $E[S_G(\lambda_G)]$.

c_X^2	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$
0	1.7059	2.2918	2.8514
0.2	1.6557	2.2092	2.7426
0.4	1.6079	2.1331	2.6367
0.6	1.5618	2.0594	2.5400
0.8	1.5240	1.9992	2.4474
1.0	1.4821	1.9260	2.3585

(a) $N = 4, b = 0.1$

Table 2: *Arrival rates λ_G for various combinations of c_X^2 and ρ*



(a) $E[B_G(\lambda_G)]$ (b) $E[S_G(\lambda_G)]$

Figure 5: $N = 4$, $b = 0.1$. *Simulation results for $\rho = 0.5, 0.7, 0.9$ as a function of c_X^2*

In addition, the simulations indicated that the α-percentiles of the response-time distribution are approximately linear in c_X^2. (A graph of the percentiles is shown in Section 6.)

Besides $N = 4$, as said we also studied a system with $N = 10$ by simulation. It turned out that the shape of the graphs shown above for $N = 4$ is identical to the shape of the graphs for $N = 10$.

Summarizing, we can conclude that the Approximation Conjecture for the $M/G/N/OCC$ system is well supported by results from simulation. The influence of the execution-time distribution on the behavior of the $M/G/N/OCC$ system can be expressed in terms of the squared coefficient of variation c_X^2.

5 Approximative Analysis of $M/G/N/OCC$

In the previous section, we saw that the behavior of an $M/G/N/OCC$ system with load ρ can be approximated reasonably accurately by a linear combination of the behavior of an $M/D/N/OCC$

and an $M/M/N/OCC$ system with the same load ρ. According to the Approximation Conjecture, the weighting coefficients of this so-called system interpolation are $1 - c_X^2$ and c_X^2.

Since fairly accurate analytic approximations for the behavior of both the $M/D/N/OCC$ and $M/M/N/OCC$ system are available (see Section 3.1 and 3.2 for a brief description), the analytic system interpolation for the $M/G/N/OCC$ system seems ready for use, except for two remaining difficulties.

The first difficulty is, that the load ρ of the $M/G/N/OCC$ queue is not known beforehand. That is, $\rho = \lambda_G E[B_G(\lambda_G)]/N$ so cannot be computed directly from the input parameters λ, b, N, and c_X^2.

The second difficulty is, that even for the simpler systems $M/D/N/OCC$ and $M/M/N/OCC$ there is no explicit formula that relates λ directly to ρ. The analyses of these systems contain an algorithm for computing $E[B(\lambda)]$, and only after $E[B(\lambda)]$ has been calculated, ρ can be obtained from $\rho = \lambda E[B(\lambda)]/N$. Thus, we cannot compute λ directly from ρ.

We solve both difficulties simultaneously by proposing a special iterative algorithm. The algorithm is based on a conjecture about the arrival rates λ_G, λ_D, and λ_M. Namely, using the relation

$$E[B_G(\lambda_G)] = (1 - c_X^2)E[B_D(\lambda_D)] + c_X^2 E[B_M(\lambda_M)],$$

which was conjectured in Section 4, and Little's law

$$N\rho = \lambda_G E[B_G(\lambda_G)] = \lambda_D E[B_D(\lambda_D)] = \lambda_M E[B_M(\lambda_M)],$$

we get the

Arrival-rate Conjecture

Let $\lambda(\rho)$ indicate that λ corresponds with a load of ρ. Then

$$\frac{1}{\lambda_G(\rho)} = (1 - c_X^2)\frac{1}{\lambda_D(\rho)} + c_X^2 \frac{1}{\lambda_M(\rho)}.$$

As an example, in Figure 6 we display some possible curves of $1/\lambda(\rho)$ as a function of c_X^2. The two endpoints of each curve correspond to the deterministic and exponential case, respectively. Note that the curves are increasing, because the $M/D/N/OCC$ system can handle larger arrival rates than the $M/M/N/OCC$ system.

In Figure 6 we also plotted an example of the point given by the input parameters $1/\lambda_G$ and c_X^2. Now the problem is to determine which ρ corresponds with this point, and which λ_D and λ_M belong to the deterministic and exponential OCC system with load ρ. So we are looking for the dashed line in Figure 6.

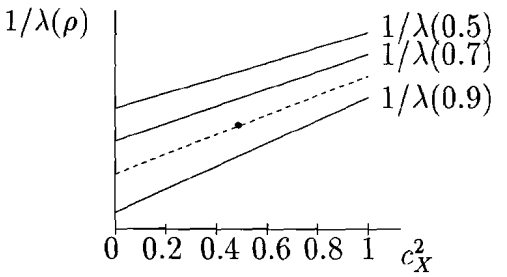


Figure 6: $1/\lambda(\rho)$ *as a function of* c_X^2

Now our analytic approximation algorithm for computing the performance of an $M/G/N/OCC$ system with arrival rate λ_G, conflict probability b, and squared coefficient of variation c_X^2 runs as follows.

Approximation Algorithm for $M/G/N/OCC$

Given λ_G, find λ_D and λ_M such that

$$\frac{1}{\lambda_G} = (1 - c_X^2)\frac{1}{\lambda_D} + c_X^2 \frac{1}{\lambda_M},$$

and such that λ_D and λ_M result in the same load ρ. (Then the $M/G/N/OCC$ system also has load ρ.) Determine the performance characteristics for the $M/G/N/OCC$ queue from the c_X^2-interpolation as stated in the Approximation Conjecture of Section 4.

So the idea is, to iteratively choose straight lines through $1/\lambda_G$ with as endpoints $1/\lambda_D$ and $1/\lambda_M$, and to compare $\rho_D(\lambda_D)$ with $\rho_M(\lambda_M)$. If $\rho_D(\lambda_D) < \rho_M(\lambda_M)$, λ_D must be raised. If $\rho_D(\lambda_D) > \rho_M(\lambda_M)$, λ_D must be lowered. As soon as $\rho_D(\lambda_D) = \rho_M(\lambda_M)$, the algorithm stops. A graphical illustration of the approximation algorithm is given in Figure 7.

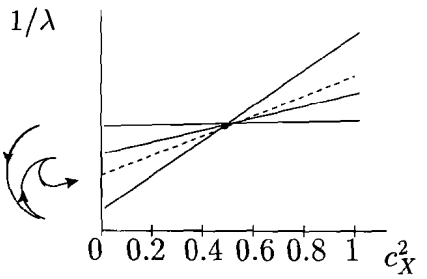


Figure 7: *The iterative approximation algorithm*

Before concluding this section, we want to stress that the quality of the approximative analysis of the $M/G/N/OCC$ queue depends largely on the quality of the analyses of the $M/D/N/OCC$ and $M/M/N/OCC$ queues that are used as endpoints in the system interpolation. The simulation study in Section 4 showed that linear interpolation is quite reasonable. However, if one of the endpoints of

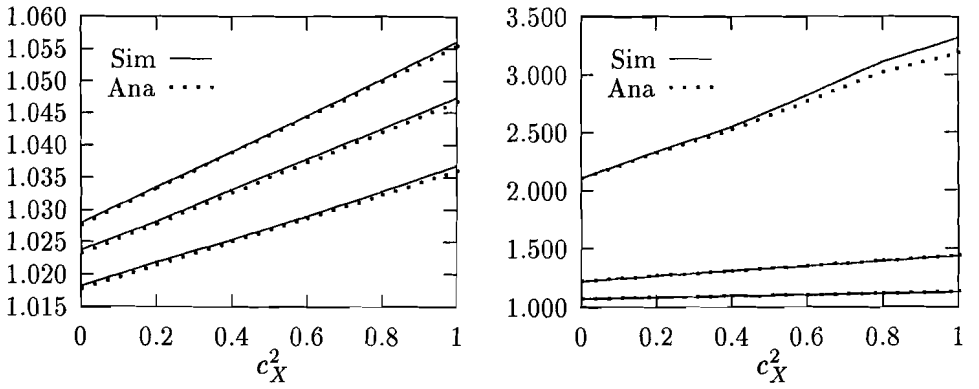
the interpolation is estimated inaccurately by the approximative analysis, this affects the approximation of all intermediate $M/G/N/OCC$ systems with the same load. So we can already say that it will be difficult to find an accurate approximation for the behavior of $M/G/N/OCC$ systems with both ρ and Nb large, because our approximation for the $M/M/N/OCC$ queue in those situations is not very good (see [13]).

6 Numerical Results

In this section, we test the approximative analysis proposed in Section 5 against simulation, for various choices of the input parameters λ_G, N, b, and c_X^2.

The number of CPUs N was taken equal to 4 and b was varied as 0.01 and 0.1. For each of these cases, we studied execution times with $c_X^2 = 0$, 0.2, 0.4, 0.6, 0.8, and 1 ($E[X] = 1$). For every combination of b and c_X^2, we considered three possibilities for the arrival rate λ_G, namely the rates for which the $M/G/N/OCC$ system has loads of 0.5, 0.7, and 0.9. These values for λ_G were already obtained in Section 4 by iterative simulation, see Table 1 and 2.

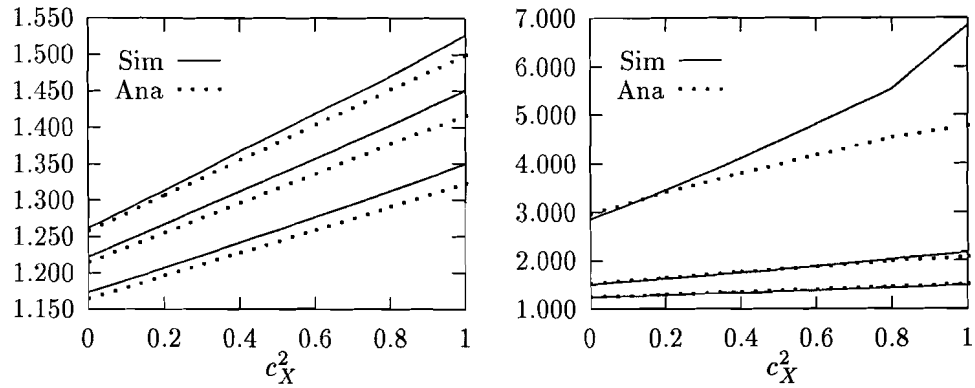
Thus, we did the analysis for exactly the same 36 systems as reported in Figure 4 and 5. Figure 8 and 9 present the analytic and simulation results together, to allow for easy comparison. The solid lines are simulation results (with loads of 0.5, 0.7, and 0.9), the dotted lines are analytic results.



(a) $E[B_G(\lambda_G)]$ (b) $E[S_G(\lambda_G)]$

Figure 8: $N = 4$, $b = 0.01$. *Analysis vs simulation with* $\rho = 0.5$, 0.7, 0.9 *for various* c_X^2

That some of the dotted analytic lines are not linear in c_X^2, while the analysis is a linear interpolation, may seem strange at first sight. However, this is simply explained by the fact that the system loads on these lines are not equal. The input parameter is λ_G and not ρ. Hence, for every choice of c_X^2, it can happen that the analysis estimates the load to be slightly different from 0.5, 0.7 or 0.9. Then the interpolation is between an $M/D/4/OCC$ and an $M/M/4/OCC$ system with this different load.



(a) $E[B_G(\lambda_G)]$ (b) $E[S_G(\lambda_G)]$

Figure 9: $N = 4$, $b = 0.1$. *Analysis vs simulation with* $\rho = 0.5, 0.7, 0.9$ *for various* c_X^2

The figures clearly show that for many of the 36 cases investigated, our approximative analysis produces quite accurate results for both $E[B_G(\lambda_G)]$ and $E[S_G(\lambda_G)]$.

For $b = 0.01$ the analysis is excellent, even when both ρ and c_X^2 are large. For $b = 0.1$, analysis and simulation of the average service time $E[B_G(\lambda_G)]$ also agree reasonably well: the discrepancy between analysis and simulation is less than 10% for all values of ρ and c_X^2 considered. Moreover, the analysis yields a very good approximation for the average response time $E[S_G(\lambda_G)]$ if the system load is 0.5 or 0.7.

Only for $b = 0.1$ and $\rho = 0.9$, the analysis of $E[S_G(\lambda_G)]$ is not accurate, but this is not surprising. It is a direct consequence of the fact that the $M/M/N/OCC$ system hasn't been satisfactorily analyzed for this combination of b and ρ (see the remark at the end of Section 5).

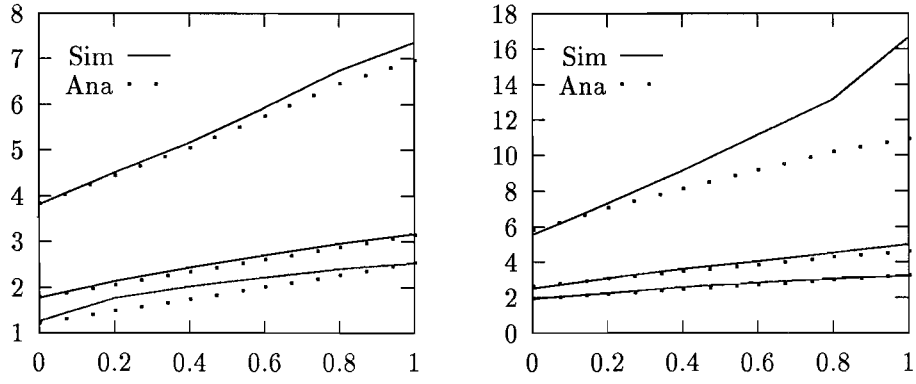
For the percentiles $\xi(\alpha)$ of the response-time distribution, the story is about the same as for average response times: the approximative analysis is very good for $b = 0.01$, and for $b = 0.1$ and $\rho \leq 0.7$. The 0.9-percentiles are shown in Figure 10 for $N = 4$ and $b = 0.01$ and 0.1, respectively.

Our conclusion from these and earlier experiments is as follows.

1. The analysis gives a good approximation for $E[B_G(\lambda_G)]$ and thus for the load ρ of the $M/G/N/OCC$ system.

2. If both ρ and Nb are large, the system becomes unstable (with very large response times) in case of exponential execution times: long execution times may lead to an enormous number of reruns. (As an indication for 'large' we suggest ρ larger than 0.8 and Nb larger than 0.3).
   In this case pure OCC is not advisable.

3. In all other cases the approximations for the average response times, as well as for the tails of the distribution, are quite good, and certainly sufficiently accurate for design decisions.



(a) $N = 4, b = 0.01$ (b) $N = 4, b = 0.1$

Figure 10: *Analysis vs simulation of* $\xi_G(0.9)$ *for* $\rho = 0.5, 0.7, 0.9$ *as a function of* c_X^2

7 Conclusions

In this paper, we studied an analytical approach for computing the performance of a real-time shared-memory database with optimistic concurrency control. The analysis provides an approximation for both the average and the distribution of the response time, and thus for the probability that deadlines are met.

We approximated the behavior of the system by a multi-server queue with a special type of feedback. The analysis of this queueing system was based on an interpolation of corresponding systems with constant and exponential execution times, which we already analyzed in [14] and [13], respectively.

The analysis provides a very accurate approximation of the response-time distribution provided ρ and Nb are not both large. The reason for the inaccuracy of the analysis in the latter case is the enormous volatility of the $M/M/N/OCC$ system under high system loads. Apart from the fact that this volatile system behavior is hard to analyze, it is also not desirable from a performance point of view, for the response-time distribution has an extremely fat tail. Ways to reduce the volatility are to postpone the execution of long transactions to a later moment in time, or to restrict the number of reruns by using a locking concurrency control algorithm for transactions in their k-th rerun. Simulation revealed that the response times then reduce drastically, yielding both better system performance and better analyzability. Such less-optimistic concurrency control algorithms are a subject for our future research.

Appendix A

In this appendix, we give the algorithm for approximating the waiting-time distribution and the response-time distribution of the $M/D/N/OCC$ queue. The interpretation of the algorithm can be found in [12].

Approximation Algorithm for $P(W \leq x)$

1. Calculate the integer m and the real number u such that $x = u + mD$ and $0 \leq u < D$.
2. Compute $r_0(u), \ldots, r_{(m+1)N-1}(u)$ from the recursion

$$r_0(u) = a_0^{-1}(u)\varphi_0, \quad r_k(u) = a_0^{-1}(u)\left(\varphi_k - \sum_{\ell=0}^{k-1} r_\ell(u) a_{k-\ell}(u)\right),$$

   where $a_k(u) = e^{-\lambda u}(\lambda u)^k/k!$.
3. Let $d_l(mD) = \binom{mN}{l} p(N)^l (1-p(N))^{mN-l}$. Then

$$P(W \leq x) = \sum_{k=0}^{N-1} r_k(u) + \sum_{k=N}^{(m+1)N-1} r_k(u)\left(1 - \sum_{l=0}^{k-N} d_l(mD)\right).$$

Conditioning on the number i seen by an arriving transaction yields the following approximation for the response-time distribution:

$$P(S \leq t) = \sum_{i=0}^{N-1} \varphi_i P(DG_{i+1} \leq t) + \left(1 - \sum_{i=0}^{N-1} \varphi_i\right) P(W + DG_N \leq t \mid W > 0),$$

with $P(W + DG_N \leq t \mid W > 0) = [P(W + DG_N \leq t) - P(W = 0)P(DG_N \leq t)]/P(W > 0)$, and with $P(W \leq x)$ computed according to the above algorithm.

Appendix B

In this appendix, we give the approximative formula for the response-time distribution $P(S \leq t)$ of the $M/M/N/OCC$ queue.

$$P(S \leq t) = 1 - \sum_{i=0}^{N-1} \prod_{n=1}^{i} \left(\frac{\lambda}{\mu_n}\right) \varphi_0 \left\{ \int_0^t (1 - e^{-\alpha_{i+1} bx})^{\lfloor \frac{t}{x} \rfloor} \mu e^{-\mu x} \mathrm{d}x + e^{-\mu t} \right\} +$$

$$- \varphi_N \mu_N \left\{ \frac{e^{-(\mu_N - \lambda)t}}{\mu_N - \lambda} + \frac{e^{-\mu t} - e^{-(\mu_N - \lambda)t}}{\mu_N - \mu - \lambda} \right\} +$$

$$- \frac{\varphi_N \mu_N}{\mu_N - \lambda} \int_0^t \left\{ \frac{1 - [e^{(\mu_N - \lambda)x}(1 - e^{-\alpha_N bx})]^{\lfloor \frac{t}{x} \rfloor - 1}}{1 - e^{(\mu_N - \lambda)x}(1 - e^{-\alpha_N bx})} e^{-(\mu_N - \lambda)(t-x)} (e^{(\mu_N - \lambda)x} - 1)(1 - e^{-\alpha_N bx}) \right.$$

$$\left. + (1 - e^{-(\mu_N - \lambda)(t - \lfloor \frac{t}{x} \rfloor x)})(1 - e^{-\alpha_N bx})^{\lfloor \frac{t}{x} \rfloor} \right\} \mu e^{-\mu x} \mathrm{d}x,$$

where $\alpha_i = (i-1)\mu_i/i$. The remaining integrals in the expression have to be evaluated numerically.

References

[1] AGRAWAL, R., CAREY, M.J., AND LIVNY, M. Concurrency control performance modeling: alternatives and implications. *ACM Transactions on Database Systems 12* (1987), 609–654.

[2] CROMMELIN, C.D. Delay probability formulae when the holding times are constant. *Post Office Electrical Engineers Journal 25* (1932), 41–50.

[3] HARITSA, J.R., CAREY, M.J., AND LIVNY, M. On being optimistic about real-time constraints. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2–4, 1990, Nashville, Tennessee* (1990), ACM Press, pp. 331–343.

[4] KIMURA, T. Approximations for multi-server queues: system interpolations. *Queueing Systems 17* (1994), 347–382.

[5] KLEINROCK, L., AND MEHOVIĆ, F. Poisson winner queues. *Performance Evaluation 14* (1992), 79–101.

[6] KUMAR, V. *Performance of concurrency control mechanisms in centralized database systems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1996.

[7] KUNG, H., AND ROBINSON, J. On optimistic methods for concurrency control. *ACM Transactions on Database Systems 6* (1981), 213–226.

[8] LITTLE, J.D.C. A proof of the queueing formula $L = \lambda W$. *Operations Research 9* (1961), 383–387.

[9] MENASCÉ, D.A., AND NAKANISHI, T. Optimistic versus pessimistic concurrency control mechanisms in database management systems. *Information Systems 7* (1982), 13–27.

[10] MORRIS, R.J.T., AND WONG, W.S. Performance analysis of locking and OCC algorithms. *Performance Evaluation 5* (1985), 105–118.

[11] PAPADIMITRIOU, C.H. *The Theory of Database Concurrency Control.* Computer Science Press, Rockville, Maryland, 1986.

[12] THE AUTHORS OF THIS PAPER Two approximations for the steady-state probabilities and the sojourn-time distribution of the $M/D/c$ queue with state-dependent feedback. Tech. Rep. COSOR 96-34, Dept. of Mathematics and Computing Science, Eindhoven University of Technology, Dec. 1996.

[13] THE AUTHORS OF THIS PAPER The response-time distribution in a real-time database with optimistic concurrency control and exponential execution times. In *Proceedings of the 15th International Teletraffic Congress - ITC 15, Washington, DC, USA, 22–27 June, 1997* (1997), V. Ramaswami and P. Wirth, Eds., North-Holland, pp. 145–156.

[14] THE AUTHORS OF THIS PAPER The response-time distribution in a real-time database with optimistic concurrency control and constant execution times. Tech. Rep. COSOR 97-07, Dept. of Mathematics and Computing Science, Eindhoven University of Technology, March 1997.

[15] TIJMS, H.C. *Stochastic Modelling and Analysis: A Computational Approach.* Wiley, Chichester, 1986.

[16] YU, P.S., DIAS, D.M., AND LAVENBERG, S.S. On the analytical modeling of database concurrency control. *Journal of the ACM 40* (1993), 831–872.