TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER

Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Performance Analysis of Real-Time Database Systems

by J.R. Haritsa

TR 92-96

Performance Analysis of Real-Time Database Systems

Jayant R. Haritsa*
Systems Research Center
University of Maryland
College Park, Maryland 20742

Abstract

During the past few years, several studies have been made on the performance of real-time database systems with respect to the number of transactions that miss their deadlines. All of these studies have used simulation models or database testbeds as their performance evaluation tools. We present, in this paper, a preliminary *analytical* performance study of real-time transaction processing. Using a series of approximations, we derive simple *closed-form* solutions to reduced real-time database models. By virtue of their simplicity, these solutions provide considerable insight into the observed performance. Although quantitatively approximate, the solutions accurately capture system sensitivity to workload parameters and yield performance bounds. Our results indicate that increased transaction slack times *degrade* performance under heavy loads for the real-time database systems considered in this study. The analysis also shows that the *absolute* sizes of transaction data sets, independent of their relationship to the database size, have a significant impact on performance. Interestingly, our approximation techniques for real-time database models are applicable to classical single-server real-time models as well, resulting in simple approximations that closely match complex exact solutions presented in the literature.

*This research was supported in part by a Systems Research Center Post-Doctoral Fellowship under NSF grant CDR-8803012.

1 Introduction.

In recent years there has been growing interest in real-time database systems, which represent a union between the hitherto separate areas of real-time systems and database systems. This interest in real-time database systems stems from the increasing number of data-intensive applications that are faced with timing requirements. These applications include aircraft control, stock trading, network management, and factory automation [Abbo88, Stan88].

In a broad sense, a real-time database system (RTDBS) is a transaction processing system that is designed to handle workloads where transactions have deadlines. The objective of the system is to meet these deadlines, that is, to process transactions before their deadlines expire. Therefore, in contrast to a conventional DBMS where the goal usually is to minimize transaction response times, the emphasis here is on satisfying the timing constraints of transactions.

Transactions may miss their deadlines in a real-time database system due to contention for physical resources (CPUs, disks, memory) and logical resources (data). During the last few years, several detailed studies [Abbo91, Hari91, Huan91] have evaluated the performance of various real-time transaction resource scheduling policies with respect to the number of missed transaction deadlines. These studies have either used simulation models [Abbo91, Hari91] or used database testbeds [Huan91] as their performance evaluation tools. To the best of our knowledge, however, no *analytical* performance studies of real-time database systems have been made so far. The lack of such studies may be attributed to the complexity of real-time database systems. Accurately modeling a real-time database system involves incorporating transaction time constraints, scheduling at multiple resources, concurrency control, buffer management, etc., and this appears to be mathematically intractable. In fact, the exact solutions to extremely simplified special cases are themselves complex (e.g. [Mitr84]).

While exact solutions appear infeasible or too complex to be of utility, we show in this paper that it is possible to derive simple *approximate* solutions to reduced models of real-time database systems. Due to their simplicity, these solutions provide considerable insight into the observed performance. Although the solutions are quantitatively approximate, they satisfactorily capture system sensitivity to workload parameters and yield limits on system performance. In essence, we are able to derive performance *trends* and *bounds*.

In this paper, we investigate the performance of real-time database systems where transactions have deadlines to the *start* of service (i.e. laxities). In our reduced model (see Figure 1.1), transactions arrive in a stream to the real-time database system, and each transaction requests the scheduler for read (shared lock) access or write (exclusive lock) access to a subset of objects in the database. A transaction that is granted access to its data *before* its laxity expires, uses the data for some period of time and then exits the system. Otherwise, the transaction is "killed", that is, it is removed from the system when its laxity expires. Our goal is to derive the steady-state fraction of input transactions that are killed (α in Figure 1.1), as a function of the workload and system parameters. We consider only data contention in our model since it is a fundamental performance limiting factor, unlike resource contention which can be reduced by purchasing more resources and/or faster resources.

Even after making several simplifying assumptions about the transaction workload process (Poisson arrivals, exponentially distributed laxities, etc.), deriving an exact solution for

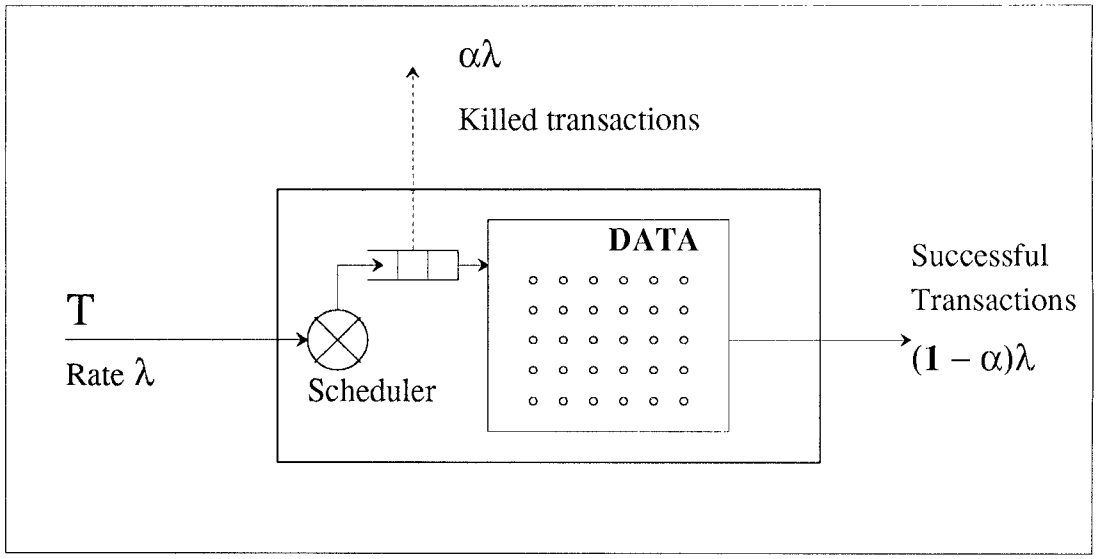

Figure 1.1: RTDBS Model

the above model appears difficult. However, using a series of approximations, we develop here a simple closed-form solution that merely involves finding the roots of a cubic equation. This approximate solution accurately captures the qualitative behavior of the RTDBS model. Further, it also provides quantitative results that are fairly close to the exact values (as determined by simulation). Taking advantage of the simplicity of the approximate solution, we derive interesting corollaries, some of which are unique to the database environment. For example, we show that increasing the laxity of tasks, which usually results in improved performance in classical real-time systems, may either improve or *degrade* RTDBS performance depending on the system operating region.

In addition to the above RTDBS results, we show that our approximation techniques can also be used to derive simple solutions for classical real-time models of single-server queues with impatient customers. Complex exact expressions have been presented for these models in the scheduling literature (e.g. [Haug80, Zhao89]). In [Haug80] the calculations involve several numerical integrations, and in [Zhao89] they involve computing incomplete gamma functions. Our solutions, however, only require evaluating the roots of cubic or quadratic equations. These approximations closely match the values provided by the exact solutions.

The remainder of this paper is organized in the following fashion: In Section 2, we formally describe our RTDBS model and the notations used in our derivations. A brief review of related work is outlined in Section 3. The approximation process and the solution to the RTDBS model are presented in Section 4. This is followed by the validation of the approximate solution in Section 5. Interesting corollaries are derived from the solution in Section 6. Then, in Section 7, the approximation techniques developed in Section 4 are used to construct simple solutions for classical real-time models. Finally, in Section 8, we summarize the main conclusions of the study and outline future research avenues.

2 Model and Notation

In our model, the database is a set O of data objects $o_1, \ldots, o_N$, where N is the number of objects in the database. Each transaction T is characterized by a 5-tuple $(A_T, L_T, O_T, M_T, P_T)$ where A_T is the transaction's arrival time; L_T is the transaction's laxity (relative to its arrival time); O_T is the set of data objects $o_i : i \in 1, \ldots, N$ that the transaction wishes to access; M_T is the desired access mode (Read or Write) vector for the objects in O_T; and P_T is the data processing time (if the transaction receives access to O_T).

The data processing model is as follows: Each transaction upon arrival informs the system scheduler of its data set (O_T) requirements and the access mode for each of these objects (M_T). If the scheduler provides the transaction access to the requested database objects before its laxity (L_T) expires, the transaction processes the data for a period P_T, and then leaves the system. Otherwise, the transaction is killed when its laxity expires.

We consider a system where transaction arrivals are Poisson with rate λ, transaction data processing times are exponentially distributed with mean $1/\mu$, and transaction laxities are (independently) exponentially distributed with mean $1/\gamma$ $(\lambda, \mu, \gamma > 0)$. We assume that the database is large, that it is accessed uniformly, and that transactions request their data sets atomically (i.e. each transaction needs simultaneous access to *all* of its data objects). We also assume that each transaction requests J data objects and that J is much smaller than N, the database size (this is usually true in practice).

In our system, the scheduler processes transactions in a FCFS manner, that is, a transaction is allowed entry into the database only if it has no data conflict with currently executing transactions *and* if all transactions that arrived prior to it have either been killed or been processed or are currently being processed. Using a FCFS policy might be considered unreasonable in a real-time environment. However, since our study is but the first step towards placing the emerging field of real-time database systems on a firmer theoretical footing, we have considered only this policy here. Further, FCFS may be the only choice when the scheduler is unaware of transaction laxities and processing times or when the scheduler is forced, from considerations of fairness, to not provide preferential treatment. A practical example of this situation is that of brokers submitting real-time buy and sell orders in a stock exchange, wherein FCFS processing may be used to maintain fairness among brokers. In our future work, we plan to investigate prioritized scheduling disciplines where transactions may be served in out-of-arrival order.

In the subsequent discussions, we use α $(0 \leq \alpha \leq 1)$ to denote the steady-state fraction of input transactions that are killed. To succinctly characterize our system configuration, we use $A/B/m/L/S$, the extended form of Kendall's notation described in [Zhao89]. (In this notation, A describes the inter-arrival process, B the service time distribution, m the number of servers, L the laxity distribution, and S represents the scheduling discipline.) For the database environment described above, the number of "servers" is not fixed, but variable depending on N, J, and the current sequence of transaction data requests. For example, multiple transactions may enter the database at the same time if their data requests do not conflict with existing locks. We will therefore use the notation N_J for the m descriptor, thereby highlighting the variability in the number of servers. With this convention, our real-time database model is represented by a $M/M/N_J/M/FCFS$ queueing system, and our goal is to characterize the α behavior of this system.

3 Related Work

Queueing systems such as $M/M/1/M/FCFS$ and $M/M/m/M/FCFS$ have been solved exactly with respect to the α metric [Haug80, Zhao89]. In [Zhao89], an elegant analysis of the $M/M/1/M/FCFS$ system was made and it was shown that the resultant solution is considerably more complex than that of the classical (non-real-time) $M/M/1/FCFS$ system. In fact, the solution involves expressions that require computation of incomplete gamma functions. A different approach to the analysis was taken in [Haug80] wherein the final solution requires several numerical integrations. Note that we cannot use these results for determining the performance of our queueing model since the number of servers in the database is variable.

Database systems where queueing is not allowed were considered in [Mitr84, Lavn84]. In these systems, a transaction that cannot receive service as soon as it arrives is immediately killed (equivalently, all transactions have zero laxity). The exact solution for this model was shown to be quite complex in [Mitr84] and approximations to the solution for large databases were presented in [Lavn84, Mitr84]. In our model, where queueing is included, the situation becomes further complicated, especially since the number of servers is variable. Therefore, deriving an exact solution appears mathematically infeasible. Moreover, even if the exact solution could be derived, the resulting expressions would probably be too involved to draw useful conclusions. In our analysis, therefore, we have sacrificed quantitative accuracy to a limited extent in order to gain computational simplicity and qualitative insights.

4 Analysis of $M/M/N_J/M/FCFS$

In this section, we present an approximate solution to the $M/M/N_J/M/FCFS$ queueing system described in Section 2. Our solution is in two parts: First, we characterize α_h, the steady-state fraction of transactions that are killed among those transactions that successfully manage to reach the *head* of the transaction wait queue. Next, we compute α_b, the steady-state fraction of transactions that are killed before they reach the head of the queue, that is, while they are in the "body" of the queue. These quantities are related to the overall α of the system by the following equation (derived by elementary flow analysis of Figure 4.1 which shows the queueing model)

$$1 - \alpha = (1 - \alpha_h)(1 - \alpha_b) \tag{1}$$

Therefore, if we are able to separately compute the "head-of-queue" and "body-of-queue" performance statistics, we can then easily derive the overall system performance. Our motivation for taking this two-step approach is to de-couple the data conflict analysis from the queueing analysis and thereby simplify the performance modeling.

In the following derivations, we refer to $\rho = \lambda/\mu$ as the system offered load, and to $\delta = \mu/\gamma$ as the normalized mean laxity (following the terminology of [Zhao89]). Further, we refer to $\xi = J/N$ as the database access ratio. For ease of explanation, we initially derive results for the case where each transaction accesses all its data objects in Write (exclusive lock) mode. Later, in Section 4.4, these results are extended to the situation where data is accessed in both Read and Write modes.

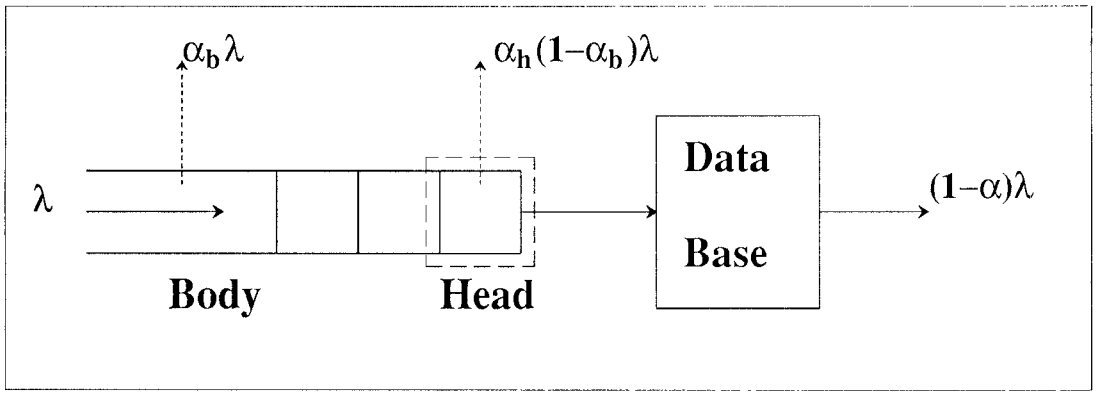

Figure 4.1: Queueing Model

4.1 Head-Of-Queue Performance

In this section, we compute α_h $(0 \leq \alpha_h \leq 1)$, the probability that a transaction which has successfully managed to reach the head of the queue is killed while waiting in this position.

Lemma 1 *The probability of a transaction at the head of the queue being killed is approximately given by*

$$\alpha_h = A(1-\alpha) \tag{2}$$

where the coefficient $A = \dfrac{\rho\xi J}{1+\delta}$.

Proof: Consider a transaction that reaches the head of the queue when k database objects are currently locked and finds that some of the data objects it requires are in this locked set (i.e. the transaction has data conflicts). The probability that this transaction is killed while waiting for the conflicting locks to be released is given by

$$\alpha_{h|k} = \sum_{i=1}^{J} P_{con|k,i}\, P_{exp|i} \tag{3}$$

where $P_{con|k,i}$ is the probability that the transaction conflicts on i of its requested J objects, and $P_{exp|i}$ is the probability that the transaction's laxity expires before the i objects on which it is conflicting are released.

We now separately consider two cases, one where k is of the same order of magnitude as J (i.e. $k \sim J$), and the other where k is much greater than J (i.e. $k \gg J$). Intuitively, we expect the former case to typically occur at low transaction arrival rates or when transactions have small laxities, while the latter case may frequently occur at high arrival rates or when transactions have large laxities.

4.1.1 Case 1: $k \sim J$

For computing $P_{con|k,i}$ we consider all possible selections wherein i of the J items in the transaction's data set are requested from the k already-locked objects and the remaining

$J - i$ items are chosen from the $N - k$ free data objects. From simple combinatorics, it is straightforward to obtain this probability as

$$P_{con|k,i} = \frac{\binom{k}{i}\binom{N-k}{J-i}}{\binom{N}{J}} \tag{4}$$

Now, since $k \sim J$ and $J \ll N$, we have $k \ll N$. Using this fact and the approximation that $\binom{n}{r} \approx \frac{n^r}{r!}$ if $n \gg r$, the above equation reduces to

$$P_{con|k,i} \approx \frac{\binom{k}{i}\binom{N}{J-i}}{\binom{N}{J}} \approx \binom{k}{i} \frac{J!}{(J-i)!} \frac{1}{N^i} \tag{5}$$

Note that this probability of conflicting on i objects is inversely proportional to N^i. Since N is large, we make the further approximation of ignoring all terms except $i = 1$ in computing $\alpha_{h|k}$. Therefore, Equation 3 is re-written as

$$\alpha_{h|k} = P_{con|k,1}\, P_{exp|1} \tag{6}$$

Substituting $i = 1$ in Equation 5 gives

$$P_{con|k,1} = \frac{kJ}{N} = k\xi \tag{7}$$

We next compute $P_{exp|1}$, which is the probability that the head waiter's remaining laxity is less than the single conflicting transaction's remaining processing time. Since transaction laxities and execution times are exponentially distributed (with parameters γ and μ, respectively), and by virtue of the memoryless property of exponential distributions, we obtain

$$P_{exp|1} = \int_0^\infty e^{-\mu t}\, \gamma e^{-\gamma t}\, dt = \frac{\gamma}{\mu + \gamma} = \frac{1}{1+\delta} \tag{8}$$

Substituting the above results for $P_{con|k,1}$ and $P_{exp|1}$ in Equation 6, we have

$$\alpha_{h|k} = k\frac{\xi}{1+\delta} \tag{9}$$

4.1.2 Case 2: $k \gg J$

We now turn our attention to the case where k is much larger than J. In this situation, the head waiter's choosing of J data items can be approximately modeled as J samplings with replacement, that is, as a sequence of Bernoulli trials [Triv82]. Since the probability of

choosing an already locked item is $\frac{k}{N}$, the probability of exactly i conflicts is approximately given by

$$P_{con|k,i} = \left(\begin{array}{c} J \\ i \end{array} \right) (\frac{k}{N})^{i} (1 - \frac{k}{N})^{J-i} \tag{10}$$

We next compute $P_{exp|i}$, which is the probability that the head waiter's laxity expires before all its i conflict locks are released. Due to the assumption of uniform access to the database and since $J \ll N$, the probability of having more than one conflict with the same transaction is small. We therefore assume that each of the i conflicts occurs with a different transaction. The cumulative distribution of the maximum of i identically-distributed exponential variables with parameter μ is given by

$$F_{max(i)}(t) = (1 - e^{-\mu t})^{i}$$

Since we are only interested in values of t that are greater than the remaining laxity of the transaction, the expression $e^{-\mu t}$ tends to 0 with increasing laxity (which is when we expect k to be much greater than J). We therefore make the approximation that

$$F_{max(i)}(t) \approx (1 - ie^{-\mu t})$$

and then obtain

$$P_{exp|i} = \int_0^\infty ie^{-\mu t} \gamma e^{-\gamma t} dt = \frac{i\gamma}{\mu + \gamma} = \frac{i}{1+\delta} \tag{11}$$

Substituting the above results in Equation 3 gives

$$\begin{aligned} \alpha_{h|k} &= \sum_{i=1}^{J} \left(\begin{array}{c} J \\ i \end{array} \right) (\frac{k}{N})^{i} (1 - \frac{k}{N})^{J-i} (\frac{i}{1+\delta}) \\ &= \frac{kJ}{N(1+\delta)} \sum_{i=1}^{J} \left(\begin{array}{c} J-1 \\ i-1 \end{array} \right) (\frac{k}{N})^{i-1} (1 - \frac{k}{N})^{J-i} \end{aligned}$$

Making the variable substitutions of $i' = i - 1$ and $J' = J - 1$, we have

$$\begin{aligned} \alpha_{h|k} &= \frac{kJ}{N(1+\delta)} \sum_{i'=0}^{J'} \left(\begin{array}{c} J' \\ i' \end{array} \right) (\frac{k}{N})^{i'} (1 - \frac{k}{N})^{J'-i'} \\ &= k\frac{\xi}{1+\delta} \end{aligned} \tag{12}$$

since the summation is identically equal to 1. Note that this final expression for $\alpha_{h|k}$ is the same as that obtained earlier for the case $k \sim J$ (Equation 9). Therefore, over the entire range of k values, we have $\alpha_{h|k} = k\frac{\xi}{1+\delta}$.

We now go on to compute α_h, the *unconditional* probability that a head-of-queue waiter is killed. Using P_k to denote the probability of k objects being locked, α_h can be expressed as

$$\alpha_h = \sum_k \alpha_{h|k} P_k = \sum_k k\frac{\xi}{1+\delta} P_k = \frac{\xi}{1+\delta} E(k) \tag{13}$$

Here, $E(k)$ is the average number of locked objects and is easily computed using Little's formula [Klei75]. The rate at which transactions obtain locks is $\lambda(1-\alpha)J$ and locks are held for a mean duration of $1/\mu$. It therefore follows from Little's formula that

$$E(k) = \frac{\lambda(1-\alpha)J}{\mu} = \rho J(1-\alpha) \tag{14}$$

Combining Equations 13 and 14, we finally obtain

$$\alpha_h = \frac{\xi}{1+\delta}\rho J(1-\alpha) = \frac{\rho\xi J}{1+\delta}(1-\alpha)$$

□

Note that in the above derivation, we made a series of approximations to finally derive a simple expression for α_h. The expression is asymptotically exact as $N \to \infty$. However, as we show in Section 5, these approximations work quite well even when N is not very large.

4.2 Body-Of-Queue Performance

In this section, we compute α_b $(0 \leq \alpha_b \leq 1)$, the steady-state probability that a transaction in the queue is killed before reaching the head of the queue, that is, while it is in the body of the queue.

Lemma 2 *The value of α_b is a unique root of the cubic equation*

$$\alpha_b^3 + B\alpha_b^2 + C\alpha_b + D = 0 \tag{15}$$

where the coefficients B, C, and D, are given by

$$\begin{aligned} B &= \frac{1}{\rho\delta}(1+\frac{1}{1+\delta}) - (1+\frac{\delta}{1+\delta}) \\ C &= (1-\frac{2}{1+\delta}) - \frac{1}{\rho\delta}(1+\frac{1}{1+\delta}) - \frac{1}{\rho^2\xi J}(1+\frac{2}{\delta}) \\ D &= \frac{1}{1+\delta} \end{aligned}$$

Over the range of valid parametric values, the equation has exactly one root in [0,1] - this is the required root.

Proof: A detailed proof of this lemma is given in Appendix A. Here, we will sketch the outline of the proof. The basic idea behind our solution is to treat the transaction wait queue *itself* as an $M/G/1$ system with the head of queue position playing the role of the "server". That is, we treat the original queue as being composed of a server and a secondary queue. As shown in Appendix A, it is possible to express the "service-time" distribution of this system (i.e. the distribution of the time spent at the head-of-queue position) with the following equation

$$f_h(t) = (1-E)u_0(t) + E(\mu+\gamma)e^{-(\mu+\gamma)t} \tag{16}$$

where $E = \rho\xi J(1-\alpha_b)(1-\alpha)$ and $u_0(t)$ is the impulse (or Dirac delta) function [Klei75].

From Equation 16, we infer that a fraction $(1-E)$ of the input transactions have a service time of zero while the remainder have an exponentially distributed service time with parameter $(\mu+\gamma)$. The transactions that have a service time of zero are those that are killed before they reach the head of the queue and those that immediately enter the database on reaching the head of the queue. The remaining transactions either enter the database after waiting for some time at the head of the queue or are killed during their wait at the head of the queue.

In [Klei75], formulas for computing the waiting time distribution of M/G/1 queues are given in terms of the service-time distribution. Substituting the service-time distribution from Equation 16 in these formulas, the cumulative distribution function of the waiting time in the body of the queue works out to

$$F_w(t) = 1 - Ge^{-(\mu+\gamma)(1-G)t} \tag{17}$$

where $G = \dfrac{\rho^2\delta\xi J}{1+\delta}(1-\alpha_b)(1-\alpha)$.

Recall that α_b is (by definition) the fraction of transactions that are killed because their laxity is smaller than their waiting time in the body of the queue. Therefore,

$$\alpha_b = \int_0^\infty (1-F_w(t))\,\gamma e^{-\gamma t}\,dt \;= \frac{G}{2+\delta-G(1+\delta)} \tag{18}$$

After substituting for G , the above equation expresses α_b in terms of the system input and output parameters. Using this equation in conjunction with Equations 1 and 2, and after some algebraic manipulations, we finally arrive at the cubic equation described in the lemma. The proof that this equation has only a single root in [0,1] is given in Appendix B.

□

An important point to note here is that the above derivation is approximate. This is because the $M/G/1$ queueing results that were used in the derivation assume independence between the task arrival process and the service time distribution. In our case, the head-of-queue "service" time distribution (Equation 16) is *dependent* on the task arrival process since it involves terms (e.g. ρ) that are a function of the arrival process. However, as we will show in Section 5, this inaccuracy does not compromise the qualitative performance behavior, and to a significant extent does not affect the quantitative behavior also. What the approximation provides in return is a simple closed-form solution that merely involves computing the roots of a cubic equation.

4.3 System Performance

In this section, we combine the results derived above for the head-of-queue and body-of-queue statistics to compute α $(0 \leq \alpha \leq 1)$, the overall fraction of killed transactions.

Theorem 1 *For the $M/M/N_J/M/FCFS$ system, the steady-state fraction of transactions that are killed is approximately given by*

$$\alpha = 1 - \frac{(1-\alpha_b)}{1 + A(1-\alpha_b)} \tag{19}$$

where α_b is obtained from Lemma 2, and $A = \frac{\rho \xi J}{1+\delta}$ is the coefficient derived in Lemma 1.

Proof: Combining the flow equation (Equation 1) and Equation 2, we obtain

$$\begin{aligned} 1-\alpha &= (1-\alpha_b)(1-\alpha_h) \\ 1-\alpha &= (1-\alpha_b)(1-A(1-\alpha)) \\ \alpha &= 1 - \frac{(1-\alpha_b)}{1+A(1-\alpha_b)} \end{aligned}$$

□

Note that it is possible to write a cubic equation in terms of α itself without going through the intermediate step of computing the roots of the cubic for α_b in Equation 15. We do not do this, however, for two reasons: First, the cubic for α has more than one root in [0,1]; an additional mechanism to identify the appropriate root is therefore required. Second, the coefficients of this cubic are cumbersome in structure.

4.4 Shared Locks

In the above derivations, it was assumed that transactions accessed all their data objects in Write (exclusive-lock) mode. The following lemma characterizes how the α performance would change if transactions accessed objects in both Read and Write modes.

Lemma 3 *Let each transaction request a fraction $\omega\,(0 \leq \omega \leq 1)$ of its J data objects in Write mode and the remainder in Read mode. Then, Lemmas 1 and 2 and Theorem 1 apply in exactly the same form except that ξ is to be replaced by $\xi\omega(2-\omega)$ in all the equations.*

Proof: In Section 4.1, we showed that the probability of the head waiter being killed when k of the N items in the database are exclusively locked is given by the equation

$$\alpha_{h|k} = k\frac{\xi}{1+\delta} = (\frac{kJ}{N})\,\frac{1}{1+\delta}$$

When shared locks are included in the modeling framework, $k = k_s + k_e$, where k_s and k_e denote the number of shared locks and exclusive locks, respectively. In addition, for each transaction, $J = J_s + J_e$, where J_s and J_e denote the requested number of shared locks and exclusive locks, respectively. The J_e exclusive lock requests can conflict with any of the k locked data objects, while the J_s shared lock requests can conflict with any of the k_e exclusively-locked data objects. Therefore, by reasoning on similar lines to that of the Write-only case, the probability of the head-waiter being killed in the Read + Write framework is determined to be

$$\alpha_{h|k} = (\frac{kJ_e}{N} + \frac{k_e J_s}{N})\,\frac{1}{1+\delta}$$

After making the substitutions $J_e = \omega J$, $J_s = (1-\omega).J$, and $k_i = \omega k$ in the above equation, we get

$$\begin{aligned} \alpha_{h|k} &= \left(\frac{k\,(\omega J)}{N} + \frac{(\omega k)\,((1-\omega)J)}{N}\right)\frac{1}{1+\delta} \\ &= k\xi\omega(2-\omega)\,\frac{1}{1+\delta} \end{aligned}$$

which is the same as the expression for the Write-only case except that ξ is replaced by $\xi\omega(2-\omega)$.
The remaining derivations follow in similar fashion.

□

When $\omega = 1$ (all data items requested in Write mode), the expression $\xi\omega(2-\omega)$ reduces to ξ, as should be expected. Conversely, when $\omega = 0$ (all data items requested in Read mode), the expression $\xi\omega(2-\omega)$ reduces to 0. Substituting this value in the equations for α results in $\alpha = 0$. This is as expected since no data conflicts occur when data is accessed only in shared mode, that is, the database behaves like an "infinite server".

From the above results, we observe that the performance of an $M/M/N_J/M/FCFS$ real-time database system is determined by $\rho, \delta, \xi, \omega$ and J. This is in contrast to classical $M/M/1/M/FCFS$ systems where the system performance is dependent only on ρ and δ [Zhao89].

5 Validation of Analysis

In this section, we compare the performance of the approximate analysis with respect to the exact solution, as determined by simulation[1]. The simulator was built using the Modula-2-based DeNet simulation language [Livn88]. In Figures 5.1 through 5.4, we plot α, the fraction of killed transactions, as a function of ρ, the system load, for different combinations of δ (the normalized mean laxity) and ξ (the database access ratio). The transaction size, J, is set to 10 in these experiments, and all data objects are requested in Write (exclusive lock) mode. Four different values of ξ, which span the range from a large-sized database to an extremely small database were considered. The chosen ξ values were 0.0001, 0.001, 0.01 and 0.1, which correspond to database sizes of 100000, 10000, 1000, and 100 respectively. Note that while we assumed that $\xi \ll 1$ in our analysis, we also evaluated the performance for larger values of ξ since we were interested in observing at what stage the analysis broke down when the assumptions were not satisfied. In our experiments, a wide range of transaction loading levels were considered such that the resultant kill fraction varied from 0 to close to 1. A long-term operating region where the kill fraction is large is obviously unrealistic for a viable RTDBS. However, exercising the system to high kill levels provides information about the system response to brief periods of stress loading.

[1] All α simulation results in this paper show mean values that have relative half-widths about the mean of less than 5% at the 95% confidence level, with each experiment having been run until at least 50000 transactions were processed by the system.

For each of the ξ settings, we evaluated the α performance for three values of δ, the normalized laxity. The selected δ values were 0.1, 1.0 and 10.0, thus covering a spectrum of transaction slack times ($\delta = 0.1$ corresponds to transaction laxities being small compared to processing times, $\delta = 1.0$ makes the laxities comparable to processing times, and $\delta = 10.0$ results in laxities that are much greater than processing times.)

In Figure 5.1, which captures the large database situation, we observe that under light loads, the analytical solution (solid lines) provides an excellent approximation to the exact solution (broken lines) for all the laxities. At heavier loads, the quantitative matching deteriorates to some extent (for the large laxity case), but the qualitative agreement is maintained throughout the entire loading range. This experiment confirms that, for large databases, the simple cubic approximation is a good estimator of system performance.

The above experiment is repeated for progressively decreasing database sizes in Figures 5.2 through 5.4. From these figures, it is clear that the approximations provide reasonably accurate performance predictions until ξ goes above 0.01. Further, even when ξ is as large as 0.1 (Figure 5.4), the *qualitative* agreement between the analysis and the exact solution remains very close. Therefore, although our analytical solution is heavily based on the assumption that the database is large, it captures system performance trends for smaller-sized databases as well.

6 Observations

In this section, we derive interesting corollaries from the analytical solutions for α that were constructed in Section 4.

6.1 Extreme Laxity Cases

We consider two extreme cases here, one where the laxity tends to 0, and the other where the laxity tends to ∞, keeping the other workload and system parameters fixed. When laxity tends to 0, transaction wait queues do not form and $\alpha_b \rightarrow 0$. Substituting $\delta = 0$ and $\alpha_b = 0$ in Equation 19 gives

$$\alpha = \alpha_h = 1 - \frac{1}{1 + \rho\xi J} \tag{20}$$

Conversely, when laxity tends to ∞, it is clear from Equation 2 that $\alpha_h \rightarrow 0$. Substituting $\delta \rightarrow \infty$ in the equation for α_b (Equation 15) and simplifying, we obtain

$$\alpha = \alpha_b = \begin{cases} 1 - \dfrac{1}{\rho\sqrt{\xi J}} & \text{if } \rho > 1/\sqrt{\xi J} \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

This equation shows that when transactions are willing to wait almost indefinitely to obtain service, they do not get killed unless the system offered load is greater than $1/\sqrt{\xi J}$. From Equation 14, this critical system load corresponds to the average number of locked objects in the database being $\sqrt{N}$ (this expression holds when all locks are exclusive; the corresponding expression for the shared lock framework is $\sqrt{\frac{N}{\omega(2-\omega)}}$).

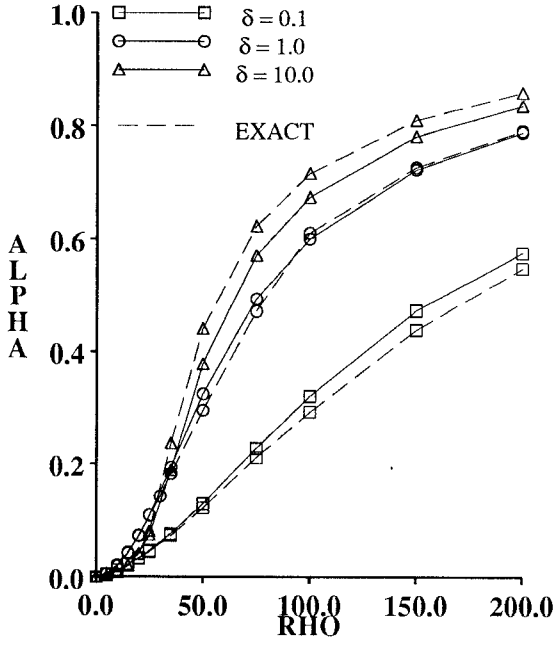

Figure 5.1: $\xi = 0.0001$

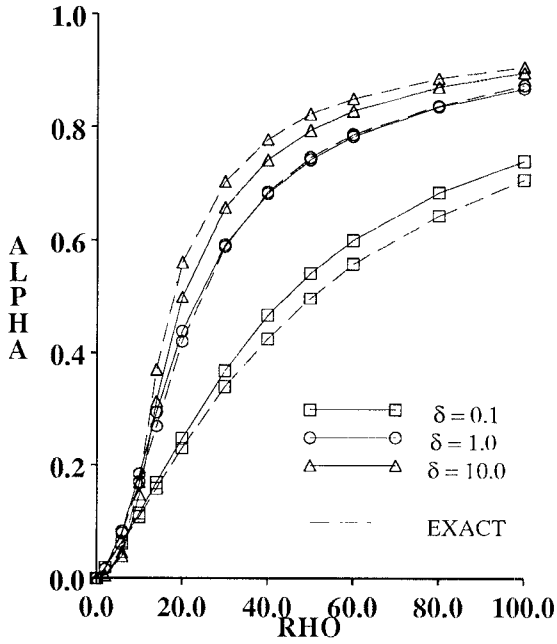

Figure 5.2: $\xi = 0.001$

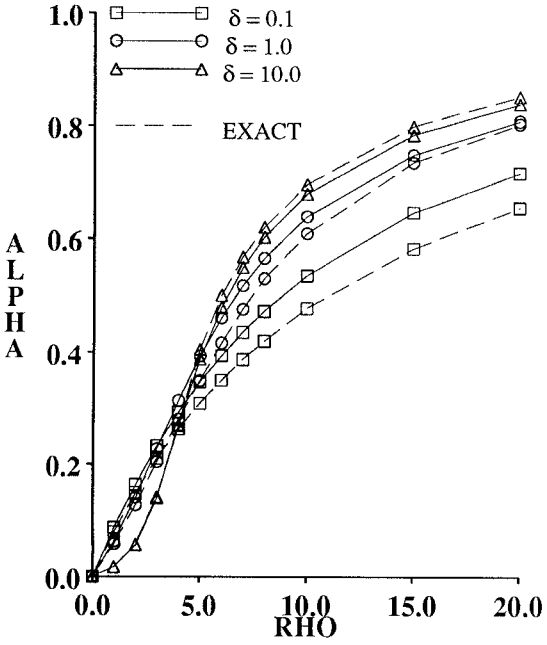

Figure 5.3: $\xi = 0.01$

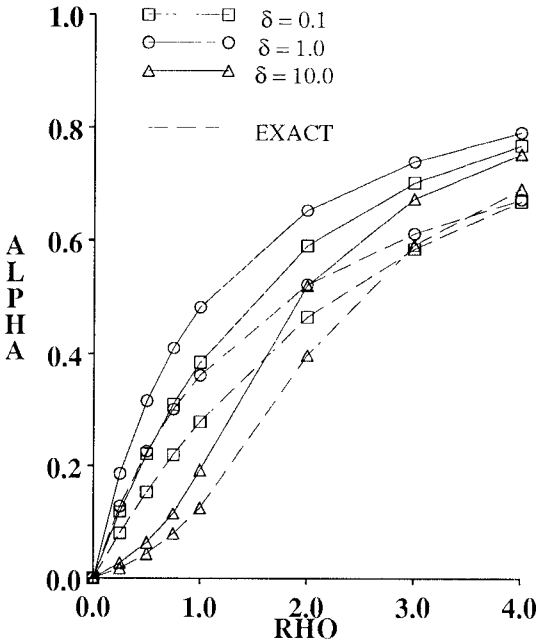

Figure 5.4: $\xi = 0.1$

6.2 Performance Crossover

A particularly interesting feature of Figures 5.1 and 5.2 is that the large laxity ($\delta = 10.0$) performance is worse than the small laxity ($\delta = 0.1$) performance over virtually the entire loading range. Further, in Figure 5.3 (it happens in Figures 5.1 and 5.2 also but is not clear due to the scale of the graph) a performance *crossover* (at $\rho = 4.0$) is clearly observed between the large laxity and the small laxity performances. This means that under light loads, large laxity results in improved performance, whereas under heavy loads, it is the other way around. Therefore, there is a critical loading point after which increased laxity can *degrade* performance. This is counter-intuitive since the expectation is that an increase in laxity should result in better performance, as observed in the corresponding classical real-time systems [Zhao89]. The reason for the difference in the database context is that transactions do not ask for generic servers, but for servers with "identity" (i.e. for specific data objects). As a result, transactions get queued up behind transactions that develop data conflicts and increased laxities result in longer queues and more conflicts. Under heavy loads, the queues become long enough that more and more transactions are killed while waiting in the queue although they have been provided with greater laxity. In short, the increased willingness to wait on the part of individual transactions is more than outweighed by the increased system queueing times that result from this willingness to wait. Therefore, increased laxity worsens performance under heavy loads for the class of real-time database systems considered in this study.

6.3 Crossover Point

In this subsection, we compute the crossover loading point beyond which the α performance with $\delta \rightarrow \infty$ becomes worse than that with $\delta = 0$. Equating the results obtained for $\delta = 0$ and $\delta \rightarrow \infty$ in Equations 20 and 21 gives

$$1 - \frac{1}{1 + \rho\xi J} = 1 - \frac{1}{\rho\sqrt{\xi J}}$$

which, after solving for ρ, leads to

$$\rho_{crossover} = \begin{cases} \dfrac{1}{\sqrt{\xi J}(1 - \sqrt{\xi J})} & \text{if } \sqrt{\xi J} < 1 \\ \infty & \text{otherwise} \end{cases} \tag{22}$$

Substituting this value of ρ back in the α equations, we obtain the α at the crossover point to be

$$\alpha_{crossover} = \begin{cases} \sqrt{\xi J} & \text{if } \sqrt{\xi J} < 1 \\ 1.0 & \text{otherwise} \end{cases} \tag{23}$$

From this expression for $\alpha_{crossover}$, it is clear that with decreasing ξ (the database access ratio), the crossover occurs at lower and lower values of α. For example, with $\xi = 0.001$ and $J = 10$, the $\alpha_{crossover}$ evaluates to 0.1. This means that from the system perspective, for loading levels that result in a kill fraction greater than 0.1, a workload of transactions that are willing to wait almost indefinitely is more difficult to handle than a workload of transactions that find only immediate service acceptable.

6.4 Performance Bounds

By evaluating the partial derivative of α w.r.t. δ in Equation 19, the following corollary is obtained:

Corollary 1: *The α performance under light loads ($\rho \rightarrow 0$) is a decreasing function of δ. Conversely, under heavy loads, ($\rho \rightarrow \infty$), the α performance is an increasing function of δ.*

Proof: The proof is provided in Appendix C.

□

From this corollary, we infer that $\delta \rightarrow \infty$ provides the lower bound on α under light loads, and the upper bound on α under heavy loads. Conversely, $\delta = 0$ provides the upper bound on α under light loads, and the lower bound on α under heavy loads. Therefore, Equations 20 and 21 provide the numerical bounds on the α performance in the light-load and heavy-load regions, respectively.

The fact that $\delta = 0$ provides the best performance under heavy loads suggests a scheduling policy where under such loads, the scheduler turns away transactions *at arrival* if they cannot be provided immediate service (even though the transactions might be willing to wait!).

We denote the loading level at which the α curve for a transaction workload with an arbitrary δ intersects the α curve corresponding to $\delta = 0$ by $\rho_{crossover}(\delta)$. Note that this crossover point can be apriori computed for any δ by using the equations of Section 4. From the graphical results shown in Figures 5.1 through 5.4 and other evaluations made by us, it appears that the following statement about the α performance behavior may hold:

> **Conjecture:** For $\rho < \rho_{crossover}(\delta)$, the α performance cannot be improved by decreasing δ. However, for $\rho \geq \rho_{crossover}(\delta)$, the lowest α is obtained with $\delta = 0$.

In order to prove the above statement, it is sufficient to show that α is a concave function of δ for all values of ρ. Theoretically, it should be possible to establish this from Equation 19 by demonstrating that the second derivative of α with respect to δ is negative. Unfortunately, the expressions for the second derivative are extremely cumbersome and we therefore currently do not have a proof of this statement.

If the above conjecture is true, it is straightforward to infer that $\rho_{crossover}(\delta)$ is an increasing function of δ. Then, the following *stronger* statement about the α performance bounds is valid:

> For $\rho < \rho_{crossover}(\infty)$, the best α performance is obtained with $\delta \rightarrow \infty$, whereas for $\rho \geq \rho_{crossover}(\infty)$, the lowest α is obtained with $\delta = 0$.

Note that this means that $\delta = 0$ and $\delta \rightarrow \infty$ provide the lower bounds on α over the *entire* loading range, and not just for the asymptotes of $\rho \rightarrow 0$ and $\rho \rightarrow \infty$ (Corollary 1).

6.5 Optimal Scheduling Policy ?

Consider the situation where the scheduler is expected to follow a strict FCFS policy with respect to processing transactions but is permitted to kill transactions *before* their laxity expires. In this situation, it is simple to see that the following scheduling policy, which we call R-FCFS (regulated FCFS), is optimal with respect to minimizing α if the conjecture discussed in the previous subsection is true:

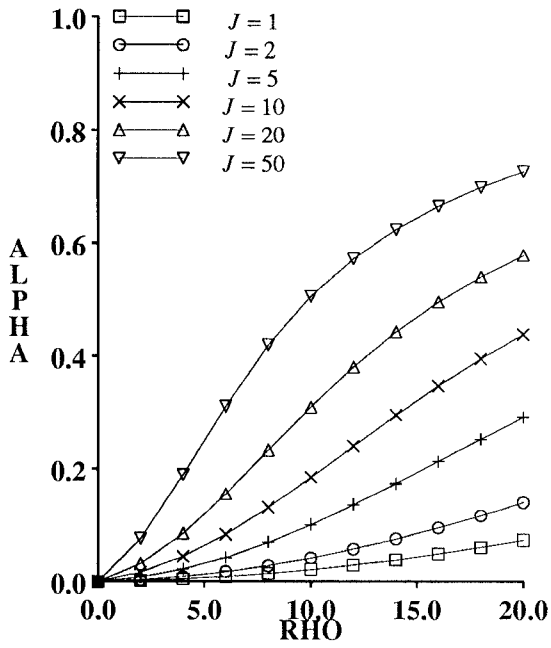

Figure 6.1: Effect of J (δ=1.0)

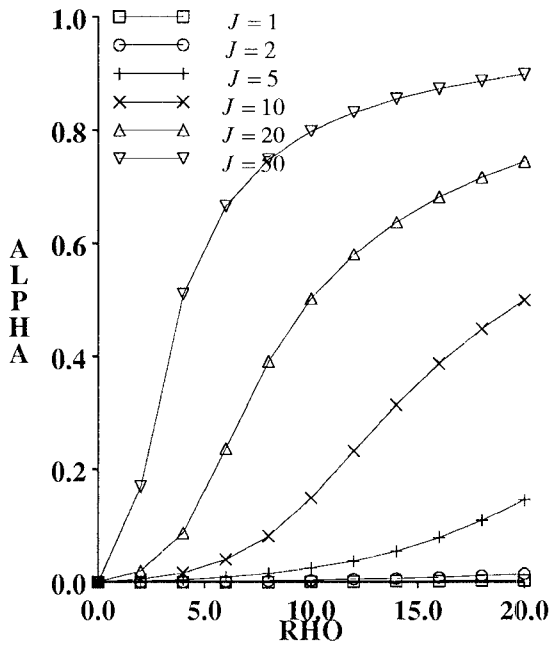

Figure 6.2: Effect of J (δ=10.0)

R-FCFS: For $\rho < \rho_{crossover}(\delta)$, follow normal FCFS scheduling. For $\rho \geq \rho_{crossover}(\delta)$, follow a policy wherein transactions are killed if they cannot be serviced immediately (i.e. treat all transactions as having zero laxity).

6.6 Effect of Transaction Size

We show in Figures 6.1 and 6.2 the effect of varying the transaction size while keeping the database access ratio fixed (i.e., the database size is scaled in proportion to the transaction size). For this experiment, we set $\xi = 0.001$ and graph α as a function of ρ for different values of J, the transaction size. In Figure 6.1, δ is set to 1.0 and in Figure 6.2, δ is set to 10.0. It is clear from these figures that the absolute value of the transaction size plays a significant role in determining system performance. This is in contrast to the classical $M/M/1$ and $M/M/1/M/FCFS$ systems where performance is determined solely by *normalized* quantities [Zhao89].

7 M/M/1/M/FCFS System

In this section, we examine how our approximate solutions, which were derived in the context of a database system, fare in the conventional real-time domain. As mentioned earlier in Section 3, exact analytical solutions exist for $M/M/1/M/FCFS$ models [Haug80, Zhao89]. The $M/M/1/M/FCFS$ queuing system can be easily mapped to our database model by setting $J = N = 1$ (i.e. every transaction requests the single object that is in the database). In Figure 7.1, the results obtained by setting $\xi = 1$ and $J = 1$ in Equation 19 are compared with the exact solution [Zhao89]. We see in this figure that the approximation is extremely

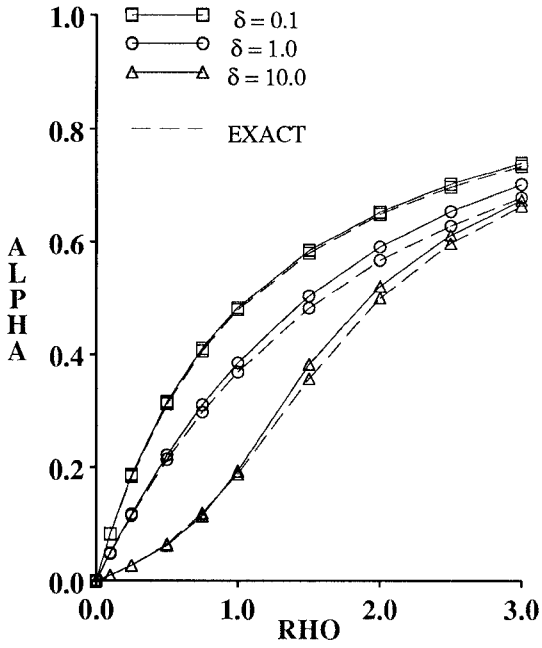

Figure 7.1: M/M/1/M/FCFS (Cubic)

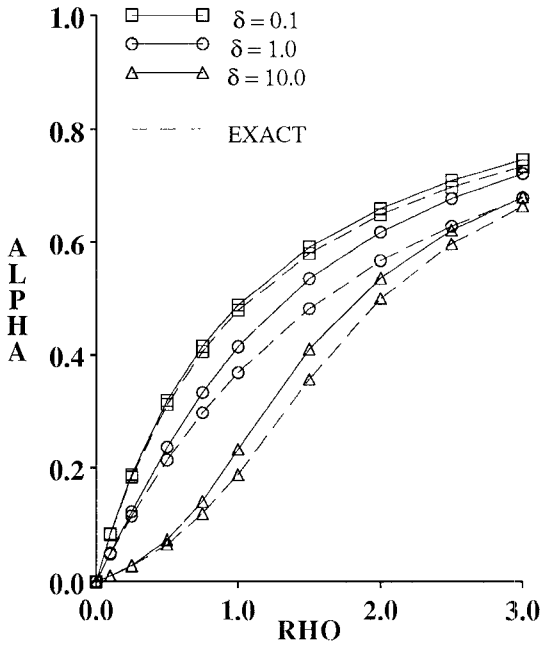

Figure 7.2: M/M/1/M/FCFS (Quadratic)

good through the entire loading range for all the laxities.

A point to note here is that the only source of approximation in the above model is in computing α_b since the expression for α_h (Equation 15) is exact when $J = N = 1$. Therefore, what we have shown is that treating an $M/M/1/M/FCFS$ system as equivalent to an $M/G/1$ queueing system feeding a server results in a simple closed-form cubic approximation that closely matches the exact solution.

For the above model, we obtain $\rho_{crossover} = \infty$ from Equation 22, which means that there is *no* performance crossover, in contrast to that seen in the database context. This agrees with the result in [Zhao89] where it was shown that $\delta \rightarrow \infty$ always provides the best performance.

7.1 Quadratic Approximation

For the $M/M/1/M/FCFS$ system, approximations that are even simpler than the cubic approximation described above can be derived. Here, instead of treating the $M/M/1/M/FCFS$ system as an $M/G/1$ system feeding a server, we consider the *entire* system to be an $M/G/1$ system. Then, using the approximation technique outlined in Section 4.2, an expression for α can be obtained with the only difference being that the derivations are modified to reflect the presence of a *real* server, rather than a pseudo-server. Following this approach, we obtain the following *quadratic* approximation:

Theorem 2 *For the M/M/1/M/FCFS system, the steady-state fraction of killed transactions is approximately given by a unique root of the equation*

$$\alpha^2 + S\alpha + T = 0 \tag{24}$$

where $S = (1 + \frac{1}{\rho})(1 + \frac{1}{\delta}) - 2$ *and* $T = -\frac{1}{\delta}$.
Over the range of valid parametric values, the equation has exactly one root in [0,1] – this is the required root.

Proof: The proof is provided in Appendices D and E.

□

In Figure 7.2, the performance of this quadratic approximation is evaluated. We observe in this figure that this approximation, while not as good as the cubic approximation, also provides a reasonably good estimate of the exact solution, especially under light loads.

8 Conclusions

In this paper, we have attempted a preliminary analytical study on the performance of real-time database systems with respect to the number of missed transaction deadlines. To the best of our knowledge, these are the first results in this area. Our goal was to provide insight into RTDBS behavioral characteristics, rather than to quantify actual system performance. To this end, we modeled the real-time database as an an $M/M/N_J/M/FCFS$ queueing system and developed an approximate closed-form solution for computing the fraction of killed transactions in this system. The solution is based on decoupling the queueing analysis from the database conflict analysis and then treating the transaction wait queue itself as an $M/G/1$ system. The solution only requires finding the roots of a cubic equation, unlike typical Markovian models where the computational complexity is often a function of the parameter values. Due to its simplicity, the approximate solution provided us with insight into the sensitivity of system performance to workload parameters and also yielded limits on performance.

Our study showed that, for medium and large-sized databases, the approximate analysis provides extremely good qualitative agreement with the corresponding simulation-derived exact results. In addition, the quantitative results are also fairly accurate, especially under light loads. For small-sized databases, the qualitative matching was retained although there was considerable deterioration in quantitative accuracy under heavy loads.

Our experiments highlighted several features that are unique to the real-time database context. The results showed that the absolute value of transaction size, independent of its relation to database size, plays a significant role in determining system performance. Therefore, we recommend that designers of real-time database applications should try to minimize the size of their transactions. Our results also showed that unlike classical real-time systems, where increased task laxity usually results in improved performance, increased transaction laxity worsens performance under heavy loads in the class of RTDBSs considered in our study. We provided a quantitative characterization of the loading level beyond which increased laxity results in degraded performance. We also showed that laxity tending to infinity provides the best performance under light loads, while laxity tending to zero is the best under heavy loads. For RTDBSs operating in the heavy load region, we recommend a scheduling policy wherein transactions are summarily rejected if they cannot be served immediately on arrival. Finally, we showed that our approximate cubic solution, in addition

to satisfactorily modeling an RTDBS, also provides a close estimate of the performance of the classical $M/M/1/M/FCFS$ real-time system. For this system, we derived an even simpler quadratic approximation that provides reasonably accurate estimates, especially under low loads. Our approximation techniques may be of interest to designers of communication networks, who frequently use similar real-time models for estimating packet dropping and call blocking probabilities.

In our study, we made several assumptions for modeling convenience: First, we assumed that the transaction scheduler uses a FCFS policy. Second, we assumed that transaction laxities and processing times are exponentially distributed. Finally, we assumed that transaction deadlines were to the start of service. We are currently working on extending the results presented here to prioritized scheduling disciplines and deterministic distributions of laxities and processing times. Developing approximation models for the case where deadlines are to the end of service and service pre-emptions ("restarts" in database terminology) are permitted appears to be a challenging research problem.

Recently, there has been work done on estimating the performance of complex queueing systems by using asymptotic results for light and heavy loads [Varm90]. In this method, the performance at any load is computed by suitably interpolating the light and heavy traffic results. It is possible that these performance estimation methods can be used to further improve the quantitative accuracy of our approximate solutions and we plan to investigate this issue in our future research.

Acknowledgements

We wish to acknowledge Lorenzo Finesso, Mrinal Ghosh, and Armand Makowski for fruitful mathematical discussions. The analysis presented in Section 4.1.2 is entirely due to Sanjoy Baruah of the University of Texas (Austin). We thank him for this assistance as also for useful comments on improving the presentation of the paper.

References

[Abbo88] Abbott, R., and Garcia-Molina, H., "Scheduling Real-Time Transactions," *ACM SIGMOD Record,* 17(1), March 1988.

[Abbo91] Abbott, R., "Scheduling Real-Time Transactions: A Performance Evaluation," *Ph.D. Thesis,* Dept. of Computer Science, Princeton University, October 1991.

[Haug80] Haugen, R., and Skogan, E., "Queueing Systems with Stochastic Time Out", *IEEE Trans. on Communications,* 28(12), December 1980.

[Hari91] Haritsa, J., "Transaction Scheduling in Firm Real-Time Database Systems," *Ph.D. Thesis,* Computer Sciences Dept., Univ. of Wisconsin, Madison, August 1991.

[Huan91] Huang, J., "Real-Time Transaction Processing: Design, Implementation, and Performance Evaluation," *Ph.D. Thesis,* Computer and Information Science Dept., Univ. of Massachusetts, Amherst, May 1991.

[Klei75] Kleinrock, L., "Queueing Systems," *Volume 1: Theory.* John Wiley & Sons, 1975.

[Lavn84] Lavenberg, S., "A Simple Analysis of Exclusive and Shared Lock Contention in a Database System," *Proc. of 1984 SIGMETRICS Conf.,* May 1984.

[Livn88] Livny, M., "DeNet User's Guide," Version 1.0., Computer Sciences Dept., Univ. of Wisconsin, Madison, 1988.

[Mitr84] Mitra, D., and Weinberger, P., "Some Results on Database Locking: Solutions, Computational Algorithms and Asymptotics," *Mathematical Computer Performance and Reliability,* Iazeolla, G., Courtois, P., and Hordijk, A. (eds.), North-Holland, 1984. 35(4), October 1988.

[Stan88] Stankovic, J., and Zhao, W., "On Real-Time Transactions," *ACM SIGMOD Record,* 17(1), March 1988.

[Triv82] Trivedi, K., *Probability & Statistics with Reliability, Queueing and Computer Science Applications.* Prentice-Hall, 1982.

[Varm90] Varma, S., "Heavy and Light Traffic Approximations for Queues with Synchronization Constraints," *Ph.D. Thesis,* Univ. of Maryland, August 1990.

[Zhao89] Zhao, W., and Stankovic, A., "Performance Analysis of FCFS and Improved FCFS Scheduling Algorithms for Dynamic Real-Time Computer Systems," *Proc. of 10th IEEE Real-Time Systems Symposium.* December 1989.

Appendix

A Proof of Lemma 2

We present here the detailed proof for Lemma 2. The first step is to compute $f_h(t)$, the service time distribution of transactions at the head-of-queue server. Both transactions that are killed before they reach the head of the queue and transactions that immediately enter the database on reaching the head of the queue have an effective service time of zero. Denoting the service time random variable by x_h, we have

$$f_h(0) = P(x_h = 0) = (\alpha_b + (1 - \alpha_b) * (1 - P_{con}))\, u_0(t) \tag{25}$$

where P_{con} is the probability that a transaction at the head of the queue has to wait due to data conflict and $u_0(t)$ is the impulse function. A quick way to compute P_{con} is to realize that it is equivalent to the head-of-queue kill fraction in a system where tasks have zero laxity. Therefore, using the result in Equation 2, we have

$$P_{con} = \alpha_{h|\delta=0} = \rho\xi J(1 - \alpha)$$

Substituting this expression for P_{con} in Equation 25, we obtain

$$\begin{aligned} f_h(0) &= (\alpha_b + (1 - \alpha_b) * (1 - P_{con}))\, u_0(t) \\ &= (1 - \rho\xi J(1 - \alpha_b)(1 - \alpha))\, u_0(t) \\ &= (1 - E)\, u_0(t) \end{aligned} \tag{26}$$

where $E = \rho\xi J(1 - \alpha_b)(1 - \alpha)$.

The transactions that do not fall into the above categories either gain entry into the database before their laxity expires or are killed while positioned at the head of the queue. The service time distribution for a transaction with remaining laxity l is given by

$$f_{h|l} = \begin{cases} \mu e^{-\mu t} & 0 < t < l \\ e^{-\mu l}\, u_0(t - l) & t \geq l \end{cases} \tag{27}$$

where the first equation corresponds to the case where the transaction's data conflict disappears before its laxity expires, and the second equation corresponds to the case where the transaction is killed.

Therefore, the unconditional pdf of the service time distribution when the service time is greater than 0 is

$$\begin{aligned} f_h(t > 0) &= \int_t^\infty (\mu e^{-\mu t} + e^{-\mu l}\, u_0(t - l))\, \gamma e^{-\gamma l}\, dl \\ &= (\mu + \gamma)\, e^{-(\mu+\gamma)t} \end{aligned} \tag{28}$$

Combining the expressions in Equations 26 and 28, the complete service time pdf is given by

$$f_h(t) = (1 - E)u_0(t) + E(\mu + \gamma)e^{-(\mu+\gamma)t}$$

Denoting the Laplace transform of the service time pdf by $H^*(s)$, we have

$$H^*(s) = 1 - \frac{Es}{s + \mu + \gamma} \tag{29}$$

Then, using the well-known $M/G/1$ results [Klei75], we compute the Laplace transform of the waiting time distribution to be

$$W^*(s) = \frac{s(1 - \rho_h)}{s - \lambda - \lambda H^*(s)} \tag{30}$$

where ρ_h is the "utilization" of the head-of-queue server.
After substituting for $H^*(s)$ in the above equation and then taking the inverse transform, we get

$$w(t) = (1 - \rho_h)u_0(t) + (1 - \rho_h)\lambda E e^{-(\mu+\gamma-\lambda E)t} \tag{31}$$

Consequently, the CDF of the waiting time is given by

$$\begin{aligned} F_w(t) &= \int_0^t w(t)\,dt \\ &= (1 - \rho_h)(1 + \frac{\lambda E}{\mu + \gamma - \lambda E}(1 - e^{-(\mu+\gamma-\lambda E)t})) \end{aligned} \tag{32}$$

The ρ_h parameter is easily computed as

$$\rho_h = \lambda \overline{x}_h = \lambda(-\frac{d}{ds}H^*(s)\Big|_{s=0}) = \lambda\frac{E}{\mu + \gamma}$$

Substituting this value of ρ_h in Equation 32 and simplifying, we have

$$F_w(t) = 1 - Ge^{-(\mu+\gamma)(1-G)t} \tag{33}$$

where $G = \dfrac{\rho^2\delta\xi J}{1+\delta}(1 - \alpha_b)(1 - \alpha)$.
Recall that α_b is the probability of a transaction being killed due to its laxity being smaller than its wait time in the body of the queue. Therefore,

$$\alpha_b = \int_0^\infty (1 - F_w(t))\,\gamma e^{-\gamma t}\,dt$$

After substituting for $F_w(t)$ from Equation 33 and evaluating the integral, the above expression reduces to

$$\alpha_b = \frac{G}{2 + \delta - G(1 + \delta)} \tag{34}$$

Substituting for G in this equation, and then solving for α_b, we have

$$\alpha_b^2 + P\alpha_b + 1 = 0 \tag{35}$$

where $P = 2 - (2+\delta)(\frac{1}{1+\delta} + \frac{1}{\rho^2 \delta \xi J(1-\alpha)})$.
From the flow equation (Equation 1) and from Equation 2, we can express α in terms of α_b as

$$\begin{aligned} 1-\alpha &= (1-\alpha_b)(1-\alpha_h) \\ 1-\alpha &= (1-\alpha_b)(1-A(1-\alpha)) \\ \alpha &= 1 - \frac{(1-\alpha_b)}{1+A(1-\alpha_b)} \end{aligned} \tag{36}$$

Substituting this expression for α in Equation 35 and making algebraic manipulations, we finally obtain

$$\alpha_b^3 + B\alpha_b^2 + C\alpha_b + D = 0$$

where B, C, and D are the coefficients given in Equation 15.
This is a cubic equation in α_b and in Appendix B it is shown that this equation has exactly one root in the range $[0, 1]$.

□

B Single Root Proof for Equation 15

Here we prove that Equation 15 has exactly one root in [0,1] over the range of valid parametric values. Let $f(\alpha_b)$ denote the LHS of Equation 15. At $\alpha_b = 0$, the value of $f(\alpha_b)$ is $\frac{1}{1+\delta}$ which is strictly positive. At $\alpha_b = 1$, the value of $f(\alpha_b)$ is $\frac{-1}{\rho^2 \xi J}(1 + \frac{2}{\delta})$ which is strictly negative. Therefore, Equation 15 has either one or all three roots between 0 and 1.

We now prove that at least one of the roots of the cubic equation must be negative. From elementary algebraic theory, we know that the product of the roots of the equation is equal to $-D = -\frac{1}{1+\delta}$. This expression is strictly negative, which means that either one or all three roots are negative.

Reconciling the above statements, it is straightforward to determine that Equation 15 has a root that is less than 0, a root that is between 0 and 1, and a root that is greater than 1.

□

C Proof of Corollary 1

The partial derivative of α w.r.t δ in Equation 19 is given by

$$\frac{\partial \alpha}{\partial \delta} = \frac{\frac{\partial \alpha_b}{\partial \delta} + \frac{\partial A}{\partial \delta}(1-\alpha_b)^2}{(1+A(1-\alpha_b))^2} \tag{37}$$

Since $A = \dfrac{\rho\xi J}{1+\delta}$, we have

$$\frac{\partial A}{\partial \delta} = -\frac{\rho\xi J}{(1+\delta)^2}$$

which tends to 0 as $\rho \to 0$.
By evaluating the partial derivative of α_b w.r.t. δ from Equation 15 and substituting $\rho \to 0$, we obtain

$$\frac{\partial \alpha_b}{\partial \delta} = -\alpha_b \frac{2}{\delta(2+\delta)}$$

Finally, the quantity $(1 + A(1-\alpha_b))^2$ tends to 1 as $\rho \to 0$.

After substituting these results in the RHS of Equation 37, it is straightforward to infer that as $\rho \to 0$, $\dfrac{\partial \alpha}{\partial \delta}$ is negative, which implies that α is a decreasing function of δ.

A derivation on similar lines shows that α is an increasing function of δ for $\rho \to \infty$.

D Proof of Theorem 2

We present here the detailed proof for Theorem 2. The first step is to compute $f_h(t)$, the service time distribution of transactions at the server. Transactions that are killed before they reach the server have an effective service time of zero, while the remaining transactions have an exponentially distributed service time with parameter μ. Denoting the service time random variable by x_h, the service time pdf is given by

$$f_h(t) = (1-E)u_0(t) + E\,\mu e^{-\mu t}$$

where $E = (1-\alpha)$.
Denoting the Laplace transform of the service time pdf by $H^*(s)$, we have

$$H^*(s) = 1 - \frac{Es}{s+\mu} \tag{38}$$

Then, using the well-known $M/G/1$ results [Klei75], we compute the Laplace transform of the waiting time distribution to be

$$W^*(s) = \frac{s(1-\rho_s)}{s - \lambda - \lambda H^*(s)} \tag{39}$$

where ρ_s is the utilization of the server.
After substituting for $H^*(s)$ in the above equation and then taking the inverse transform, we get

$$w(t) = (1-\rho_s)u_0(t) + (1-\rho)\lambda E e^{-(\mu-\lambda E)t} \tag{40}$$

Consequently, the CDF of the waiting time is given by

$$\begin{aligned} F_w(t) &= \int_0^t w(t)\,dt \\ &= (1-\rho_s)(1 + \frac{\lambda E}{\mu - \lambda E}(1 - e^{-(\mu-\lambda E)t}) \end{aligned} \tag{41}$$

Using Little's formula, the ρ_s parameter is easily computed as

$$\rho_s = \frac{\lambda(1-\alpha)}{\mu}$$

Substituting this value of ρ_s in Equation 41 and simplifying, we have

$$F_w(t) = 1 - Ge^{-\mu(1-G)t} \tag{42}$$

where $G = \rho(1-\alpha)$.
Since α is the probability of a transaction being killed due to its laxity being smaller than its wait time in the queue,

$$\alpha = \int_0^\infty (1 - F_w(t))\, \gamma e^{-\gamma t}\, dt$$

After substituting for $F_w(t)$ from Equation 42 and evaluating the integral, the above expression reduces to

$$\alpha = \frac{G}{1 + \delta - G\delta} \tag{43}$$

Substituting for G in this equation, and then solving for α, we have

$$\alpha^2 + S\alpha + T = 0$$

where S and T are the coefficients given in Equation 24.
This is a quadratic equation in α and in Appendix E it is shown that this equation has exactly one root in the range $[0, 1]$.

□

E Single Root Proof for Equation 24

Here we prove that Equation 24 has exactly one root in [0,1] over the range of valid parametric values. Let $f(\alpha)$ denote the LHS of Equation 24. At $\alpha = 0$, the value of $f(\alpha)$ is $-\frac{1}{\delta}$ which is strictly negative. At $\alpha = 1$, the value of $f(\alpha)$ is $\frac{1}{\rho}(1+\frac{1}{\delta})$ which is strictly positive. Therefore, Equation 15 has exactly one root between 0 and 1.

□