TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

The response time distribution in a multi-processor database with single queue static locking

Citation for published version (APA):
Bodlaender, M. P., Sassen, S. A. E., Stok, van der, P. D. V., & Wal, van der, J. (1995). *The response time distribution in a multi-processor database with single queue static locking*. (Memorandum COSOR; Vol. 9540). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/1995

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 08. Feb. 2024

Eindhoven University
of Technology

Department of Mathematics
and Computing Science

Memorandum COSOR 95-40

The Response Time Distribution in a Multi-Processor Database with Single Queue Static Locking

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

Eindhoven, November 1995
The Netherlands

Eindhoven University of Technology
Department of Mathematics and Computing Science
Probability theory, statistics, operations research and systems theory
P.O. Box 513
5600 MB Eindhoven – The Netherlands

Secretariat: Main Building 9.15 or 9.10
Telephone: 040-247 4272 or 040-247 3130
E-mail: wsbscaro@win.tue.nl or wscosor@win.tue.nl
Internet: http://www.win.tue.nl/win/math/bs/cosor.html

ISSN 0926 4493

The Response Time Distribution in a Multi-Processor Database with Single Queue Static Locking [1]

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

Department of Mathematics and Computing Science
Eindhoven University of Technology

November 1, 1995

Abstract

A transaction scheduling mechanism is designed for a shared-memory, multi-processor database system. The scheduler used is a variant of static locking, adapted for real-time and more than one processor.

It is assumed that transactions arrive according to a Poisson process, execution times of transactions are independent and exponentially distributed, and all transactions use the same number of data items. The system is then represented as a Markov model. A steady state is derived from this model. By examining the path through the system of a single transaction, a recursive relation that describes all moments of a transaction's response time is derived. The moments are obtained from this relation with dynamic programming.

The response time distribution is approximated by fitting a distribution to the first two moments. Simulation shows that this approximation gives excellent results.

1 Introduction

To profit from the increase in CPU power that parallel computer architectures [Hwa93] offer, transactions on databases should be executed concurrently. However, concurrent execution can destroy the consistency of the database, if transactions are incorrectly scheduled. Transactions are only allowed to execute concurrently if the effect is equivalent to a sequential execution of the same transactions. The theory of serializability is described in [Vid91].

In real-time systems [Sta95],[TC77],[eDM83], jobs are scheduled in a different way. While the schedule in a database has to maximise the throughput of transactions, real-time schedules must guarantee that each job is completed before a certain deadline. Soft real-time systems are allowed to miss some deadlines, when the system is overloaded. In hard real-time systems, all deadlines have to be met. We investigate soft real-time systems in this paper.

Real-time databases combine the scheduling constraints from both databases and real-time systems. Scheduling mechanisms have to deal with database consistency and with transaction deadlines. New schedulers must be constructed, as real-time schedulers do not guarantee database consistency, and database schedulers often have poor real-time performance.

Analysis of database schedulers [YDL93],[MW85],[KM92], [Tay87] has been restricted to throughput-analysis. The probability that a transaction meets its deadline cannot be derived from the throughput alone.

In this paper we analyse the Single-Queue, Static-Locking (SQSL) scheduler, that schedules transactions in a multi-processor, shared-memory database. The SQSL scheduler is an adaptation of static locking, and is well suited for analysis. The SQSL scheduler does not use information about deadlines, and the real-time performance drops to zero when the system is overloaded. In [Bod95] several other versions of the SQSL scheduler are presented that

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

have a much better real-time performance under high system loads. These schedulers do use deadline information, however an analysis of their response time distribution is more complex.

Our analysis uses a Markov model (see for instance [Tij94]) to capture the essential behaviour of the system. The mean and variance (and higher moments) of the response time of a transaction are derived. By fitting a distribution to these moments, an excellent approximation is obtained for the probability that a transaction meets its deadline.

2 Specification of the system and its scheduler

The hardware consists of n independent CPUs that execute transactions. CPUs have access to a shared memory, where the entire database is stored. No disks are attached to the database.

When transactions arrive at the system they are transferred to the shared memory. We assume that this takes negligible time. CPUs retrieve transactions from the shared memory to execute them. A transaction is executed by a single CPU, so CPUs need not communicate with each other during the execution of transactions. Further, we assume that the shared-memory is large enough to store the entire database and all waiting transactions.

The SQSL scheduler Transactions are scheduled using the Single-Queue, Static-Locking strategy: all transactions are handled in a first-come, first served (FCFS) manner. Transactions are allowed to execute if no data conflicts with already executing transactions occur. A data conflict occurs if two transactions need to access the same data item (we do not distinguish between reading and writing data items).

Transactions that are not executed immediately on arrival are stored in a queue. When transaction t that is first in the queue has a data conflict with an executing transaction, t must wait. Moreover, because waiting transactions cannot be overtaken by transactions that arrive later (FCFS), all other transactions in the queue must wait as well.

3 Markov model of the system

The main-memory database is modeled as a Markov chain. We assume that the arrival of transactions is a Poisson process with parameter λ. Furthermore, execution times of transactions are independent and exponentially distributed with rate μ. Up to n transactions can be executing at the same time, and the queue is unbounded.

We assume that the database stores a fixed number d of data items. Each transaction accesses a data items. All items have an equal probability of being accessed.

The states Under the above assumptions, the system state is completely described by the tuple (i, j), where i is the number of executing and j the number of waiting transactions.

When the number of executing transactions is lower than the number of available CPUs ($i < n$) and the number of waiting transactions is positive ($j > 0$), the first transaction in the queue has a data conflict with at least one executing transaction. If all CPUs are executing transactions, it is unknown whether the first transaction has a data conflict. These observations (that are in fact extra pieces of information that are incorporated into the state description) lead to the following definition of the states (i, j).

$i = 0$ and $j = 0$:	The system is empty.
$1 \leq i < n$ and $j \geq 0$:	i mutually non-conflicting transactions are executing, and j transactions are waiting, of which the first (if any) has a conflict with at least one of the i executing transactions.
$i = n$ and $j \geq 0$:	i mutually non-conflicting transactions are executing and j transactions are waiting.

Some probabilities Let $B(i)$ be the probability that a transaction has a data conflict with one or more out of i executing transactions. If transaction t at the head of the queue has a data conflict with at least one of i executing transactions, $B(i-1 \mid i)$ is the probability that t still has a data conflict with at least one of the remaining $i-1$ executing transactions, after one of the i executing transactions has left. Conflict probabilities $B(i)$ and $B(i-1 \mid i)$ are:

$$B(i) = 1 - \binom{d-ai}{a} \Big/ \binom{d}{a} \text{ and } B(i-1 \mid i) = \frac{B(i-1)}{B(i)}.$$

These equations hold as long as $a(i+1) < d$, which is the case for all realistic purposes. The expression for $B(i-1 \mid i)$ was derived by applying Bayes' formula $B(i-1,i) = B(i-1 \mid i) \cdot B(i)$. Here $B(i-1,i)$ is the probability that a transaction conflicts with at least one out of $i-1$ transactions and also conflicts with at least one out of i transactions; the $i-1$ transactions are a subset of the i transactions. $B(i-1,i)$ equals $B(i-1)$ and the result follows.

As the complements of $B(i)$ and $B(i-1 \mid i)$ are often used, we define $A(i) = 1 - B(i)$ and $A(i-1 \mid i) = 1 - B(i-1 \mid i)$.

The Markov property The processing of transactions can be described by a continuous time Markov chain with state descriptor (i,j). This follows from the exponential (thus memoryless) inter-arrival and execution times, the fixed number of items used by each transaction, and the fact that all items have an equal probability of being accessed. The future state of the system depends on the current state (i,j) and not on the past states: the Markov property holds.

Transitions of the Markov model We analyse what state transitions are possible in the model. First, transactions arrive at the system with rate λ. If there are no waiting transactions, at least one CPU is free, and i transactions are executing, with probability $B(i)$ the arriving transaction is blocked and with probability $A(i)$ it is allowed to execute:

$$(i,0) \rightarrow (i,1) \text{ with rate } \lambda B(i) \text{ if } i < n.$$

$$(i,0) \rightarrow (i+1,0) \text{ with rate } \lambda A(i) \text{ if } i < n.$$

If the number of waiting transactions j is greater than zero, or no CPU is available ($i = n$), the arriving transaction enters the queue:

$$(i,j) \rightarrow (i,j+1) \text{ with rate } \lambda \text{ if } i = n \text{ or } j > 0.$$

Second, if $i > 0$ transactions are executing, a transaction finishes its execution at rate $i\mu$. If the queue is empty, a finished transaction is not replaced:

$$(i,0) \rightarrow (i-1,0) \text{ with rate } i\mu \text{ if } i > 0.$$

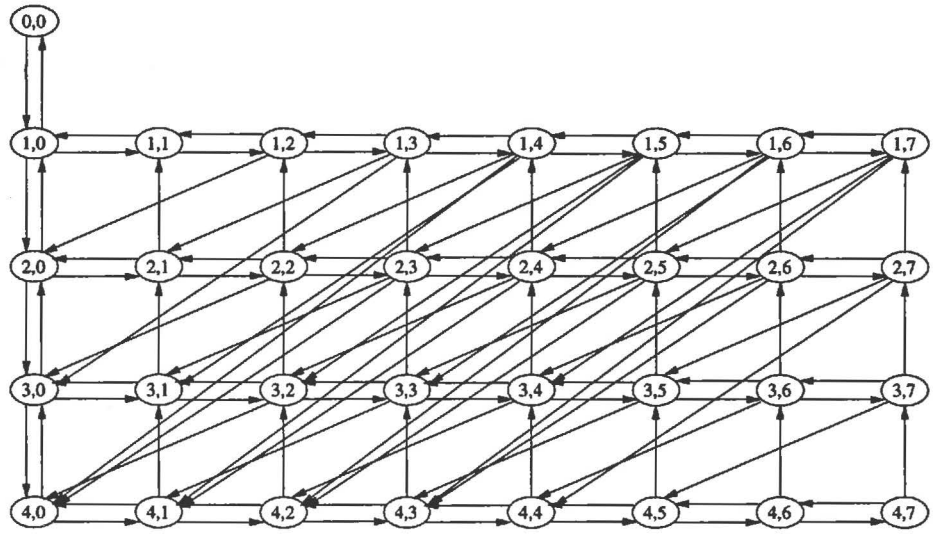


Figure 1: A MODEL OF STATIC QUEUEING FOR $n = 4$

If at least one transaction is waiting and $i = n$, with probability $B(n-1)$ the first transaction in the queue remains blocked at a transaction completion epoch. With probability $A(n-1)$ the first transaction is not blocked and can start executing:

$$(n, j) \rightarrow (n-1, j) \text{ with rate } n\mu B(n-1) \text{ if } j > 0.$$

$$(n, j) \rightarrow (n, j-1) \text{ with rate } n\mu A(n-1) \text{ if } j > 0.$$

If $j > 0$ and $i < n$ just before a transaction completes execution, with probability $B(i-1 \mid i)$ the first transaction f in the queue remains blocked:

$$(i, j) \rightarrow (i-1, j) \text{ with rate } i\mu B(i-1 \mid i) \text{ if } i < n \text{ and } j > 0.$$

With probability $A(i-1 \mid i)$ f begins execution. Now if a CPU is still available, and the queue is not empty, the transaction r that is next in line is available for execution. With probability $B(i)$ it is blocked:

$$(i, j) \rightarrow (i, j-1) \text{ with rate } i\mu A(i-1 \mid i)B(i) \text{ if } i < n \text{ and } j > 1.$$

With probability $A(i)$, r is allowed to execute. Now if yet another CPU is available, and the queue is still not empty, a third transaction is available for execution. With probability $B(i+1)$ it is blocked:

$$(i, j) \rightarrow (i+1, j-2) \text{ with rate } i\mu A(i-1 \mid i)A(i)B(i+1) \text{ if } i < n-1 \text{ and } j > 2.$$

With probability $A(i+1)$ the transaction executes. As long as there is still a CPU available and the new first transaction in the queue does not conflict with the transactions in execution, the scheduler admits a new transaction to a free CPU. The transitions that can arise and their rates are included in the following, summarizing expression. When $j > 0$, a departure

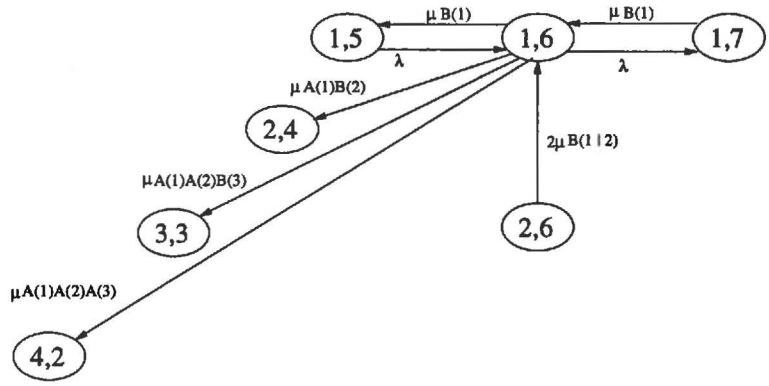


Figure 2: TRANSITIONS TO AND FROM (1,6) IF $n = 4$

can cause the following state transitions

$(i,j) \rightarrow (i-1+k, j-k)$

with rate $n\mu B(n-1)$ if $i = n$ and $k = 0$
with rate $n\mu A(n-1)$ if $i = n$ and $k = 1$
with rate $i\mu B(i-1 \mid i)$ if $i < n$ and $k = 0$
with rate $i\mu A(i-1 \mid i) \prod_{m=0}^{k-2} A(i+m)$ if $i < n$ and $k = \min\{j, n-i+1\} > 0$
with rate $i\mu A(i-1 \mid i) \prod_{m=0}^{k-2} A(i+m)B(i-1+k)$ if $i < n$ and $0 < k < \min\{j, n-i+1\}$.

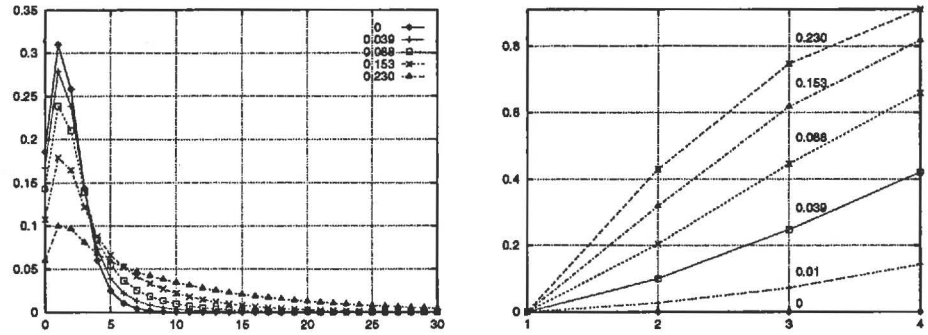
The convention is used that $\prod_{m=0}^{\ell} = 1$ if $\ell < 0$.

Figure 1 shows the possible transitions when $n = 4$, and queues are at most 7 transactions long. Observe that the system is cyclic, even and odd states can be defined (by counting the distance to (0,0)). All transitions are between an even and an odd state. Figure 2 gives a close-up of the transitions to and from (1,6), with their intensities.

4 Steady-state distribution

Let vector π denote the steady-state distribution of the Markov model described above. Then $\pi(i,j)$ is the probability that in the long run the system is in state (i,j). The steady state distribution is used in section 5 to compute the moments of the response time.

Deriving the steady state by truncating the state space Balance equations can be derived from the expressions in the previous section, by applying the "rate out of state (i,j) = rate into state (i,j)" principle. A transition matrix Q is constructed by combining all transitions defined above. Solving the balance equations (the system $Q\pi = 0$) gives the steady-state distribution π of the Markov chain.



(a) Steady states

(b) CPU waste because of blocking

Figure 3: STEADY STATES AND CPU USAGE FOR $n = 4$ AND VARIOUS a

Transition matrix Q is of infinite size (the number of states is unbounded), which makes it difficult to solve the balance equations. We computed the steady-state probabilities numerically from the balance equations by truncating the state space (and thus bounding the size of Q) at a sufficiently high number J of waiting transactions. The probability that transactions arrive in states where $\geq J$ transactions are waiting should be negligible (smaller than some tolerance value ε). Therefore, J is related to parameters n, λ, μ, a and d.

The matrix-geometric approach instead of truncation The steady-state probabilities can be computed without truncating the state space, by using the matrix-geometric approach ([Neu81]). Let level j represent all states with j waiting transactions, and define π_j as the vector $(\pi(0,j)\ldots\pi(n,j))$. Then steady-state distribution π_j has the geometric form:

$$\pi_j = \pi_1 R^{j-1} \quad \text{for } j \geq 1.$$

Matrix R is the unique solution to a matrix equation of order $n+1$. This is the equation $\sum_{k=0}^{n+1} R^k A_k = 0$ where the A_k's are specific $n \times n$ sub-matrices of transition matrix Q. Matrix R can be solved numerically by successive substitution (starting with $R = 0$).

Steady-state vector π_0 is also determined by R. Unfortunately, for most choices of n an explicit expression for R cannot be obtained, because of the high order of the matrix equation. So even when Neuts's matrix-geometric approach is used, the steady-state probabilities can only be computed numerically.

Drawing conclusions from the steady state In Figure 3(a), the steady-state distributions of the total number of transactions in the system are drawn for different conflict probabilities. The parameters used are $\lambda = \frac{5}{3}$, $\mu = 1$ and $d = 100$. The number a of data items used by a transaction was varied between 0 and 5 to obtain the different conflict probabilities. The figure shows that high conflict probabilities result in longer queues, as some transactions have to wait because of a conflict.

In Figure 3(b) the effect of blocking on CPU usage has been plotted. Depicted is the percentage of time that $< m$ CPUs are executing transactions, given that there are $\geq m$

transactions in the system, for $m = 1$ to 4. Clearly, if the conflict probability is 0, transactions never wait if a CPU is available. When conflicts become more frequent, the usefulness of the third and fourth CPU drops dramatically. Therefore, in systems where the conflict probability is high, adding CPUs will hardly increase the performance of the system.

The steady state of the system gives information about the throughput, average queue-lengths and CPU activity. If execution times differ substantially, knowledge of the average execution time is not sufficient to guarantee that deadlines are met. Information about the distribution of the response time is needed.

5 The response time distribution

The distribution of the response time S of a transaction is completely described by the moments of the response time. We aim to find $E[S^k]$, the k-th moment of S, for $k \geq 1$.

The average response time $E[S]$ of a transaction can be computed using Little's rule $E[L] = \lambda E[S]$. Here $E[L]$ is the average number of transactions in the system and is obtained from the steady-state probabilities: $E[L] = \sum_{i+j>0}(i+j)\pi(i,j)$.

For $E[S^k]$ with $k > 1$ no direct relation between $E[S^k]$ and $E[L]$, $E[L^2]$ up to $E[L^k]$ is known for systems like this. To derive an expression for $E[S^k]$ we define a recursive relation that depends on $E[S^l]$ $(l < k)$, and depends on moments of exponentially distributed stochastic variables. This process is described below.

A recursive relation for the response time We follow the path of an arbitrary transaction through the model, from arrival to departure. With a 'path' we mean the states that are reached during the presence of the transaction under consideration. Tuple $[i,j]$ describes the situation where i transactions are in execution and $\geq j$ transactions are waiting in the queue. The tuple (i,j) refers to the system state as defined before. Define

$$S_{[i,j]} = \text{the time until a transaction } t \text{ leaves the system, when } i \text{ transactions are executing, and } j-1 \text{ transactions are ahead of } t \text{ in the queue.}$$

If $j = 0$, the transaction under consideration is in execution. When the system is in state (i,j) after an arrival, $S_{[i,j]}$ is the response time of the newly arrived transaction.

Important is the observation that $S_{[i,j]}$ does not depend on transactions that arrive at the system after the transaction under consideration. This follows from the property of the Single-Queue, Static-Locking scheduler: transactions waiting in the queue cannot be overtaken.

The consequence of this observation is that arrivals of other transactions need not be considered when $E[S^k_{[i,j]}]$ is computed. Let X_i be the time till the next departure when i transactions are executing (X_i is exponentially distributed with rate $i\mu$). Let $p_{[i,j][m,\ell]}$ be the probability that the next departure leads to a state with m transactions in execution and $\ell - 1$ transactions present in the queue ahead of the transaction under consideration. Then

$$S_{[i,j]} = X_i + S_{[m,\ell]} \qquad \text{with probability } p_{[i,j][m,\ell]} \text{ for all } [m,\ell].$$

This is a recursive relation, as $m + \ell = i + j - 1$. Therefore, the moments of $S_{[i,j]}$ can be computed from the moments of $S_{[m,\ell]}$ with $m + \ell < i + j$. Once a transaction is in execution, its service time is exponentially distributed with mean $1/\mu$. Thus the boundary condition for

the recursion is $S_{[i,0]} = X$ for all $i > 0$, where X is exponentially distributed with parameter μ.

Let $a_{(r,\ell)(i,j)}$ be the probability that a transition to state (i,j) is caused by an arbitrary transaction t that sees state (r,ℓ) on arrival. An expression for t's response time S is found by conditioning on state (r,ℓ) and by using the PASTA [Wol82] property:

$$S_{[i,j]} \qquad \text{with probability} \sum_{(r,\ell):i+j=r+\ell+1} \pi(r,\ell) a_{(r,\ell)(i,j)}.$$

Deriving the moments of the response time The moments of the response time are derived directly from the recursive relation. Two important rules are used to find $E[S^k]$ for $k \geq 1$:

- **Choice.** The transaction follows path l with k-th moment $E[S_l^k]$, or path m with k-th moment $E[S_m^k]$. The probability that path l is taken is p. Then

$$E[S^k] = pE[S_l^k] + (1-p)E[S_m^k].$$

- **Addition.** The transaction first follows path l with duration S_l, followed by path m with duration S_m. Then

$$E[S^k] = E[(S_l + S_m)^k].$$

Based on these rules, the moments of S can be found using dynamic programming.

Example Suppose $n > 2$ and we need the second moment of the response time of transaction t that is first in the queue while two transactions are in execution (situation $[2,1]$). First, t must wait until a transaction leaves. This time is represented by random variable X_2 which is exponentially distributed with parameter 2μ. After the departure, t remains blocked with probability $B(1 \mid 2)$. Otherwise t begins execution. Using both choice and addition rule, the expression becomes:

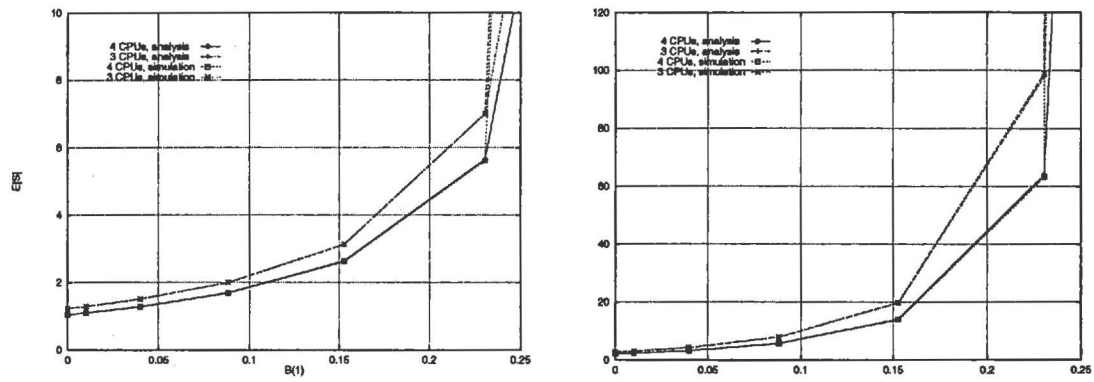
$$\begin{aligned} E[S_{[2,1]}^2] &= A(1 \mid 2)E[(X_2 + S_{[2,0]})^2] + B(1 \mid 2)E[(X_2 + S_{[1,1]})^2] \\ &= A(1 \mid 2)(E[S_{[2,0]}^2] + 2E[S_{[2,0]}]E[X_2]) + \\ &\quad B(1 \mid 2)(E[S_{[1,1]}^2] + 2E[S_{[1,1]}]E[X_2]) + E[X_2]^2. \end{aligned}$$

To find $E[S_{[2,1]}^2]$, the first and second moment of X_2, $S_{[2,0]}$ and $S_{[1,1]}$ must be known. In general, for $E[S_{[i,j]}^k]$, moments $E[S_{[m,\ell]}], \ldots, E[S_{[m,\ell]}^k]$ with $m + \ell < i + j$ and the first k moments of exponentially distributed variables are needed. Now the second moment of the response time of an arbitrary customer is (using the choice rule)

$$E[S^2] = \sum_{(r,\ell)} \pi(r,\ell) \sum_{(i,j):i+j=r+\ell+1} a_{(r,\ell)(i,j)} E[S_{[i,j]}^2].$$

6 Approximating the response time distribution by fitting

Schassberger proved that each positive stochastic variable can be approximated arbitrarily well by a weighted sum of independent exponentially-distributed variables (see [Sch93]). We



(a) First moment

(b) Second moment

Figure 4: ANALYSIS AND SIMULATION FOR $n = 3$ AND $n = 4$

used Schassberger's result to find a mixture of exponentially distributed variables that has the same moments of S. The way this mixture is chosen influences the quality of the approximation. Denote the stochastic variable corresponding to the chosen mixture by $\hat{S}$. Then $P(S \leq x)$, the probability that a transaction meets its deadline, is approximated by $P(\hat{S} \leq x)$. We say the distribution of $\hat{S}$ is fitted to the moments of S.

We used the two-moment fit as described in [Tij94]. The fitting procedure is not given here for reasons of brevity, but it can fit a distribution to any combination of $E[S]$ and $E[S^2]$.

7 Simulation compared to fitting

A simulation of the system has been programmed, and several test-runs were made. Again, $\lambda = \frac{5}{3}$ and $\mu = 1$ was used. In Figure 4, the simulation and analysis results for $E[S]$ and $E[S^2]$ are shown. It is clear that the truncation of the state space results in inaccuracies when the system is not stable.

For $n = 4$ we used moments $E[S]$ and $E[S^2]$ from our analysis to approximate $P(S \leq x)$

	$P(S \leq 1)$		$P(S \leq 3)$		$P(S \leq 5)$	
$B(1)$	Fit	Sim.	Fit	Sim.	Fit	Sim.
0	0.61	0.61	0.95	0.95	0.99	0.99
0.010	0.59	0.59	0.94	0.94	0.99	0.99
0.039	0.54	0.54	0.90	0.90	0.98	0.98
0.088	0.45	0.45	0.83	0.83	0.95	0.95
0.153	0.32	0.32	0.68	0.68	0.85	0.85
0.230	0.16	0.17	0.41	0.42	0.59	0.60

Table 1: RESULTS FOR THE RESPONSE TIME DISTRIBUTION

for $x = 1$, 3, 5 and 7. Conflict probability $B(1)$ was varied from 0 to 0.230, corresponding to $a = 0$ to $a = 5$ in a database with $d = 100$. We also estimated $P(S \leq x)$ by simulation. The results of both analysis and simulation are given in Table 1. The simulated values are the midpoints of a 95% confidence interval with a width smaller than 0.02. Table 1 shows that the fitting procedure gives an excellent approximation of the response time distribution.

8 Concluding remarks

The straightforward scheduling approach of Single Queue Static Locking allows for a thorough analysis. It proved possible to give an exact analysis of all moments of the response time of a transaction. To our knowledge, analyses of database schedulers were always restricted to the mean response time. Approximation of the response time distribution by fitting gave promising results, even when only two moments were used.

Alternatively, the recursive relation defined in section 5 can be used to construct the Laplace-Stieltjes transform of the response time. Numerical inversion of this transform (see [AW92]) also yields an approximation of the response time distribution.

Analysis was possible because of the specific nature of the SQSL scheduler, combined with important assumptions that enabled us to use a memoryless model. Further research could be directed at weakening some of the assumptions we made (such as the assumption about the fixed number of data items used by a transaction), or to extend the analysis for more elaborate versions of the SQSL scheduler. The real-time behaviour of the SQSL scheduler improves significantly when deadline information is used by the scheduler. It would be useful to analyse these better schedulers.

References

[AW92] J. Abate and W. Whitt. The fourier-series method for inverting transforms of probability distributions. *Queueing Systems 10, pp. 5-88*, 1992.

[Bod95] M.P. Bodlaender. Single queue static locking. *to be published*, 1995.

[eDM83] (ed.) D.A. Mellicamp. *Real Time Computing with applications to data acquisition an control.* van Nostrand Reinhold, 1983.

[Hwa93] K. Hwang. *Advanced Computer Architecture: Parallelism, Scalability, Programmability.* McGraw-Hill, Inc., London, 1993.

[KM92] L. Kleinrock and F. Mehović. Poisson winner queues. *Performance Evaluation 14, 79-101*, 1992.

[MW85] R.J.T. Morris and W.S. Wong. Performance analysis of locking and occ algorithms. *Performance Evaluation 5, 105-118*, 1985.

[Neu81] M. Neuts. *Matrix-Geometric Solutions in Stochastic Models.* The Johns Hopkins University Press, Baltimore, 1981.

[Sch93] R. Schassberger. *Warteschlangen.* Springer-Verlag, Berlin, 1993.

[Sta95] J.A. Stankovic. Implications of classical scheduling results for real-time systems. *Computer, Vol. 28, No. 6, pp. 16-25*, 1995.

[Tay87] Y.C. Tay. *Locking Performance in Centralized Databases.* Academic Press, Inc., 1987.

[TC77] D. Tebbs and G. Collins. *Real Time Systems, management and design.* McGraw-Hill, 1977.

[Tij94] H.C. Tijms. *Stochastic Models, an Algorithmic Approach.* John Wiley & Sons, Chichester, 1994.

[Vid91] K. Vidyasankar. Unified theory of database serializability. *Fundamenta Informaticae XIV*, 1991.

[Wol82] R.W. Wolff. Poisson arrivals see time averages. *Operations Research 30, pp. 223-231*, 1982.

[YDL93] P.S. Yu, D.M. Dias, and S.S. Lavenberg. On the analytical modeling of database concurrency control. *Journal of the ACM pp. 831-872*, 1993.

List of COSOR-memoranda - 1995

Number	Month	Author	Title
95-01	January	M.J.A. van Eenige I.J.B.F. Adan J.A.C. Resing J. van der Wal	Periodic versus exhaustive service in a multi-product production center
95-02	January	F.P.A. Coolen P. van der Laan	On Indifference Zone Selection with a Preference Threshold
95-03	February	E.E.M. van Berkum P.M. Upperman	Some new designs for quantitative factors
95-04	February	B. Pauwels E.E.M. van Berkum	Approximate and exact designs for incomplete quadratic models
95-05	February	R. Mantri & A. Saberi Z. Lin A.A. Stoorvogel	Output Regulation for Linear Discrete-Time Systems to Input Saturation
95-06	February	M.L.J. Hautus	Observability of saturated systems with an offset
95-07	February	A.A. Stoorvogel A. Saberi	Continuity properties of solutions to H_2 and H_∞ Riccati equations
95-08	March	J.L. van den Berg E.B. Diks J.A.C. Resing J. van der Wal	A fluid flow model of an ATM traffic shaper
95-09	March	J.L. van den Berg E.B. Diks J.A.C. Resing J. van der Wal	The change of traffic characteristics in ATM networks 2*
95-10	March	R. Lioce C. Martini	Heuristic Methods for Machine Scheduling Problems with Processor Sets: A Computational Investigation
95-11	March	I.J.B.F. Adan J.A.C. Resing	A note on a fluid queue driven by an $M/M/1$ queue

Number	Month	Author	Title
95-12	March	M.J.A. van Eenige I.J.B.F. Adan J.A.C. Resing J. van der Wal	Periodic service with working overtime and producing to stock in a multi-product production center
95-13	April	H.J.C. Huijberts J.H. van Schuppen	Routing Control of a Motorway Network – A Summary
95-14	April	P. van der Laan C. van Eeden	On selecting the best of two normal populations using a loss function (Revised version of 93-15)
95-15	May	K. Aardal C.P.M. van Hoesel	Polyhedral Techniques in Combinatorial Optimization
95-16	May	O. Goldschmidt D.S. Hochbaum C.A.J. Hurkens G. Yu	Approximation for the k-Clique Covering Problem
95-17	May	F.W. Steutel	A curious implication of Spitzers identity
95-18	May	J. Verrijdt I.J.B.F. Adan A.G. de Kok	A trade off between emergency repair and inventory investment
95-19	June	C.A. van Eijl	A polyhedral approach to the delivery man problem
95-20	June	P. van der Laan F.P.A. Coolen	An Overview of a Generalization in Statistical Selection
95-21	July	J.A. Hoogeveen S.L. van de Velde	Earliness-tardiness scheduling around almost equal due dates
95-22	July	J.A. Hoogeveen S.L. van de Velde	Scheduling by positional completion times
95-23	July	H.P. Stehouwer E.H.L. Aarts J. Wessels	Multi-layered perceptrons for on-line lot sizing (extended abstract)
95-24	July	J.M. van den Akker C.A.J. Hurkens M.W.P. Savelsbergh	A Time-Indexed Formulation for Single-Machine Scheduling Problems: Branch-and-Cut

Number	Month	Author	Title
95-25	August	P. van der Laan	Statistical selection: A way of thinking!
95-26	August	P. van der Laan C. van Eeden	On using a loss function in selecting the best of two gamma populations in terms of their scale parameters
95-27	August	F. Groot C.A.A.M. Withagen A. de Zeeuw	Open-loop von Stackelberg equilibrium in the cartel-versus-fringe model
95-28	August	C.A. van Eijl C.P.M. van Hoesel	On the discrete lot-sizing and scheduling problem with Wagner-Whitin costs
95-29	September	H. Kellerer T. Tautenhahn G.J. Woeginger	Approximability and Nonapproximability Results for Minimizing Total Flow Time on a Single Machine
95-30	September	J.H.J. Einmahl	Poisson and Gaussian approximation of weighted local empirical processes
95-31	September	E.B. Diks A.G. de Kok	Transshipments in a divergent two-echelon network using the consistent appropriate share rationing policy
95-32	September	C.A.A.M. Withagen M. Toman	Cumulative pollution with a backstop
95-33	September	N.V. Shakhlevich	Unit-time Open-shop Scheduling Problems with Symmetric Objective Functions
95-34	September	C.A.J. Hurkens S.R. Tiourine	Upper and lower bounding techniques for frequency assignment problems
95-35	September	J.M. van den Akker J.A. Hoogeveen S.L. van de Velde	Parallel machine scheduling by column generation
95-36	October	Z.G. Zhang R.G. Vickson M.J.A. van Eenige	Optimal two-threshold policies in an $M/G/1$ queue with two vacation types
95-37	October	S. Sevastianov	Nonstrict vector summation in multi-operation scheduling
95-38	October	V.G. Deĭneko G.J. Woeginger	Long-Chord-Free and Fence-Free Tours for the Travelling Salesman Problem
95-39	November	J.A. Hoogeveen A.P.A. Vestjens	Optimal on-line algorithms for single-machine scheduling

Number	Month	Author	Title
95-40	November	M.P. Bodlaender S.A.E. Sassen P.D.V. van der Stok J. van der Wal	The Response Time Distribution in a Multi-Processor Database with Single Queue Static Locking