TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

The M/G/infinite queue with OCC

Citation for published version (APA):
Sassen, S. A. E., & Wal, van der, J. (1997). *The M/G/infinite queue with OCC*. (Memorandum COSOR; Vol. 9718). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/1997

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. Aug. 2022

Eindhoven University
of Technology

**Department of Mathematics
and Computing Sciences**

Memorandum COSOR 97-18

The $M/G/\infty$ queue with OCC

by

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

Eindhoven, October 1997
The Netherlands

The $M/G/\infty$ queue with OCC

Simone Sassen and Jan van der Wal

Abstract

The $M/G/\infty$ queue with Optimistic Concurrency Control (OCC) is a model for transaction processing in a real-time database. Transactions arrive according to a Poisson process and require some generally distributed execution time. The number of servers is unlimited so execution of an arriving transaction always starts immediately. However there is a risk that after completion of a transaction all work turns out to be useless because another transaction has overwritten one or more data-items that were used. Thus, the total service time of a transaction consists of a stochastic number of runs, dependent on the load of the system. In this study we develop an approximation for the distribution of the total service time, and test the approximation against simulation. Although in practice the number of servers is never unlimited, this study is very useful to judge the value of having a large number of servers in OCC systems.

1 Introduction

In database systems one of the service disciplines to support parallelism is Optimistic Concurrency Control (OCC). Under OCC, services can always start, but there is the risk that after completion of a transaction all work turns out to be useless because another transaction has overwritten one or more data-items that were used. In that case, the transaction is invalidated and its execution must be repeated from scratch. Thus, under OCC, the total service time of a transaction consists of a number of runs. How many runs are needed depends on how many other transactions are executing at the same time and on the amount of data contention in the system.

In a number of previous papers (SASSEN and VAN DER WAL [1997a, 1997b]) we studied OCC systems with a *finite* number of servers. However, to understand some of the asymptotic characteristics of OCC systems, it is very useful to investigate the behavior of an *infinite* server OCC system. This study will clearly show the limited value of adding servers to cope with an increase of the transaction arrival rate.

The paper is organized as follows. In Section 2 we describe the model. Section 3 contains an approximate analysis of the mean and the distribution of the total service time in an $M/G/\infty$ queue with OCC. In Section 4 we compare the analysis with a simulation of the model, for deterministic and exponentially distributed run times. In Section 5 we analyze a slightly different OCC-mechanism called Broadcast OCC, and compare its performance to the so-called Pure OCC-mechanism that is considered in Section 2 to 4. Section 6 contains some concluding remarks.

2 The Model

The model we consider is the following. Transactions arrive according to a Poisson process with rate λ . The execution time X needed for one service run of a transaction has a general distribution function $F(x)$. We assume that transactions read all data-items they need at the start of their execution. The values of the data-items are stored in a local buffer of the CPU (server) where the execution takes place. The actual execution then comprises doing calculations with these data-items.

If a transaction is successful and completes without being invalidated by one of the other transactions it commits and leaves the system. The data-items changed locally by the transaction are then updated globally (in main memory). When the transaction commits it invalidates any of the other transactions in the system with probability b.

Whether a transaction has been invalidated is checked only at the end of the servicing. If it has been invalidated it just restarts. We assume identical repeat, that is, if the first service run of a transaction takes x time units then so do all the repetitions. When a transaction is restarted, all data-items it needs are read anew from main memory. We call this *complete data refreshment.*

The number of servers is ∞ , so upon arrival a transaction immediately gets a server.

3 The Analysis

For the time being we assume that all jobs (transactions) that enter eventually leave again. So the average number of departures per time unit is equal to the number of arrivals λ.

Consider a transaction T that requires x time units of servicing. Every transaction that commits during T's execution invalidates T with probability b. Thus, if the random variable N_x denotes the number of commits during T's run of length x, the probability that the run is successful is[1]

$$E\left[(1-b)^{N_x}\right] = \sum_{n=0}^{\infty} P(N_x = n)(1-b)^n. \tag{1}$$

The problem is, to determine the distribution of N_x. The number of commits during T's run depends on the time since the previous commit and on the remaining execution times of the other transactions in the system. We are not very optimistic about the chances of an exact analysis of the distribution of N_x. Therefore, we make the following approximative assumption.

Departure Assumption
The departure process of successfully committing transactions is a Poisson process with rate λ.

So we assume that N_x is Poisson distributed with mean λx. Recall that in the ordinary $M/G/\infty$ queue the departure process is indeed Poisson (see e.g. DOOB [1953], p. 405-406, KENDALL [1964] and NEWELL [1966]). In the OCC model there is dependence between the departure epochs

[1] $E[Z]$ is the expectation of random variable Z

of the various transactions, so it is unlikely that the departure process will still be Poisson. In the following extreme case it is clear that the departure process is not Poisson.

Example
Consider the case with $b = 1$. Assume all service times are deterministic and equal to 1. Then if a transaction completes, all transactions in the system are invalidated. Thus during the next time unit no departures are possible. Hence the interdeparture time is at least 1.

However we conjecture that in all practical situations (i.e., with b small, say $b \leq 0.2$) the Departure Assumption is very close to reality.

Now let us return to transaction T of length x. Substitution of $P(N_x = n) = e^{-\lambda x}(\lambda x)^n/n!$ in (1) yields a success probability for T of $e^{-\lambda bx}$. (Another way to find this success probability is to realize that under the Departure Assumption the time between two invalidations is exponentially distributed with parameter λb.) Hence the total service time of T, including reruns, is a geometric sum of periods of length x, the mean of which is $xe^{\lambda bx}$. We will call this total time the generalized service time.

Result 3.1 *The mean generalized service time of a transaction of duration x, denoted by $g(x)$, satisfies*

$$g(x) = xe^{\lambda bx}.$$

So for any finite x, any λ and any b the mean generalized service time is finite. This justifies the assumption made at the beginning of this section that the departure process of the transactions has the same rate as the arrival process.

However, the overall mean generalized service time may be no longer finite even if the mean execution time (the mean of X) is finite.

Result 3.2 *The overall mean generalized service time g satisfies*

$$g = \int_0^\infty xe^{\lambda bx}\,\mathrm{d}F(x)\,,$$

with F the execution time distribution.

Further the mean number of active servers L is given by

$$L = \lambda g\,.$$

If the execution time is deterministic with mean 1 then the overall mean generalized service time g_D is given by

$$g_D = e^{\lambda b}.$$

If the execution time is exponentially distributed with mean 1 then the overall mean generalized service time g_M is finite only if $\lambda b < 1$, in which case we have

$$g_M = \frac{1}{(1-\lambda b)^2}\,.$$

Besides an expression for the (overall) mean generalized service time, we can also derive an expression for the distribution function of the (overall) generalized service time. Since under the Departure Assumption the generalized service time of a transaction of duration x is a geometric sum of periods of length x, the following result is immediate.

Result 3.3 *The distribution function of the generalized service time of a transaction of duration x, denoted by $G_x(t)$, satisfies*

$$G_x(t) = 1 - (1 - e^{-\lambda b x})^{\lfloor \frac{t}{x} \rfloor}.$$

(The notation $\lfloor y \rfloor$ denotes the largest integer smaller than or equal to y.)

Conditioning on the value of x yields Result 3.4.

Result 3.4 *The distribution function $G(t)$ of the overall generalized service time G satisfies*

$$G(t) = F(t) - \int_0^t (1 - e^{-\lambda b x})^{\lfloor \frac{t}{x} \rfloor} \, \mathrm{d}F(x),$$

with F the execution time distribution.

A way to measure the *real-time* performance of a system is to compute the percentage of transactions that meets its deadline. If $\alpha\%$ of the transactions meets deadline t, we call the system (t, α)-efficient. From Result 3.3 (or 3.4) we easily obtain

Result 3.5 *There exists a finite maximum value λ^* of the arrival intensity λ for which an $M/G/\infty$ system with OCC and conflict probability b is still (t, α)-efficient.*

4 Numerical Results

In this section we consider two cases, deterministic and exponential execution time, in somewhat more detail. For various choices of the input parameters λ and b, results produced by simulations are compared with results from the approximative analysis for these two cases.

The quality of the approximation for the distribution of the generalized service time depends on the quality of the Departure Assumption. Therefore, we studied the actual departure process of the $M/G/\infty$ queue with OCC by simulation. According to the Departure Assumption, the interdeparture time (i.e., the time that elapses between two consecutive commits) is exponentially distributed with parameter λ. This implies that the coefficient of variation of the interdeparture time (i.e., the standard deviation of the interdeparture time divided by the mean) equals 1. Further, the assumption says that consecutive interdeparture times are independent.

We studied the departure process for the cases of deterministic and exponential execution times, for $b = 0.01$, 0.1, and 0.2, respectively. In all simulations, the average interdeparture time was indeed equal to $1/\lambda$.

For $b = 0.01$, the coefficient of variation of the interdeparture time was 0.99 for all cases of λ considered, both for deterministic and for exponential execution times. Moreover, the simulations showed that subsequent interdeparture times were practically uncorrelated and independent. Hence, for $b = 0.01$ the Departure Assumption is almost exact. For $b = 0.1$ and 0.2, we found that the interdeparture times are less exponential (i.e., the coefficient of variation is less than 1) and subsequent interdeparture times become more (negatively) correlated. Whether this causes a serious error in the approximative analysis is seen below.

Table 1 and 2 compare analysis with simulation for deterministic and exponential execution times, respectively. The input parameters are the arrival intensity λ and the conflict probability b. The average execution time is taken equal to 1. The tables show the average number of active servers L, the overall mean generalized service time g, and the tail probabilities $P(G > 1)$, $P(G > 2)$, and $P(G > 5)$.

		L_D		g_D		$P(G_D > 1)$		$P(G_D > 2)$		$P(G_D > 5)$	
λ	b	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim
20.0	0.01	24.43	24.43	1.22	1.22	0.18	0.18	0.033	0.033	0.0002	0.0002
40.0	0.01	59.7	59.7	1.49	1.49	0.33	0.33	0.11	0.11	0.0039	0.0039
80.0	0.01	178.0	178.5	2.23	2.23	0.55	0.55	0.30	0.30	0.051	0.051
2.0	0.10	2.44	2.45	1.22	1.22	0.18	0.18	0.033	0.033	0.0002	0.0002
4.0	0.10	5.97	6.00	1.49	1.50	0.33	0.33	0.11	0.11	0.0039	0.0043
8.0	0.10	17.80	18.13	2.23	2.27	0.55	0.56	0.30	0.31	0.051	0.055
1.0	0.20	1.22	1.23	1.22	1.23	0.18	0.18	0.033	0.034	0.0002	0.0002
2.0	0.20	2.98	3.02	1.49	1.51	0.33	0.34	0.11	0.11	0.0039	0.0047
4.0	0.20	8.90	9.26	2.23	2.31	0.55	0.57	0.30	0.32	0.051	0.059

Table 1: *Analysis versus simulation for deterministic execution times*

All simulations were run until 4 million transactions committed. The simulation results in the tables are accurate up to the last digit shown[2]. Numerical integration was used to do the analysis for $P(G > t)$ in the case of exponential execution times.

Analysis versus Simulation

Since the analysis of the generalized service time G only depends on the product λb and not on λ and b separately, the results the analysis produces for G in the upper, middle and lower part of

[2]That is, a simulated value of 2.45 means, that the 95% confidence interval lies inside [2.44, 2.46]. Similarly, a simulated value of 112 means that the 95% confidence interval lies inside [111, 113]. The only exceptions are the values in italics in Table 2, which have a wider confidence interval. More about those values later in this section.

Table 1 (Table 2) are the same. However, the simulation results for G in the three parts of Table 1 (Table 2) differ slightly. This indicates that the system behavior is not completely characterized by λb but depends also on λ and b separately. In general we see that as b becomes larger (keeping λb fixed), the simulated values of g and $P(G > t)$ become larger (whereas the analytic values remain unchanged). Apparently, the analysis overestimates the success probability of transactions when $b = 0.1$ or 0.2.

		L_M		g_M		$P(G_M > 1)$		$P(G_M > 2)$		$P(G_M > 5)$	
λ	b	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim
20.0	0.01	31.3	31.3	1.56	1.56	0.40	0.40	0.20	0.20	0.056	0.056
40.0	0.01	111	112	2.78	2.79	0.43	0.43	0.25	0.25	0.10	0.10
80.0	0.01	2000	*1100*	25	*14*	0.48	0.48	0.33	0.33	0.19	0.19
2.0	0.10	3.12	3.14	1.56	1.57	0.40	0.40	0.20	0.20	0.056	0.057
4.0	0.10	11.1	11.3	2.78	2.82	0.43	0.43	0.25	0.25	0.10	0.11
8.0	0.10	200	*151*	25	*19*	0.48	0.48	0.33	0.33	0.19	0.19
1.0	0.20	1.56	1.57	1.56	1.57	0.40	0.40	0.20	0.20	0.056	0.057
2.0	0.20	5.6	5.8	2.78	2.92	0.43	0.43	0.25	0.25	0.10	0.11
4.0	0.20	100	*89*	25	*22*	0.48	0.48	0.33	0.34	0.19	0.19

Table 2: *Analysis versus simulation for exponential execution times*

The more deterministic the commit process and the higher b, the larger is the error made by the analysis. The analysis then overestimates the success probability so underestimates the mean and tail of G. Nevertheless, in both Table 1 and 2 (and in all other investigated cases) we see that the influence of the separate parameters λ and b on the distribution of the generalized service time is not large.

The tables clearly show that the analysis produces an excellent approximation for the mean and distribution of the generalized service time, both with deterministic and with exponential execution times. Only in the cases of Table 2 where the simulation results are printed in italics, the analysis appears to be (very) inaccurate. A detailed investigation of these cases led to the surprising conclusion that the discrepancy between analysis and simulation is not caused by a bad analysis, but by a bad (that is, too short) simulation!

In order to explain why the simulations that produced the values in italics were too short, we consider the case of exponential execution times with $\lambda = 80$ and $b = 0.01$. According to the analysis, $L_M = 2000$ and $g_M = 25$. Simulation of 4 million transactions gives $L_M \approx 1100$ and $g_M \approx 14$. To understand this difference, let us consider the set of transactions with an execution time of 15 or more. A simulation run of 4 million transactions contains on the average only 1.2 transactions with an execution time of 15 or more. In equilibrium, however, the number of transactions of length 15 and more is $\lambda \int_{15}^{\infty} x e^{\lambda b x} e^{-x} \, \mathrm{d}x$, so nearly 400. So the simulation should have had a duration of at least 400 times 4 million transactions. But this also means that the long transactions will experience extremely long delays, and that OCC for long transactions is disastrous.

So the system with 4 million simulated transactions is still far away from statistical equilibrium (the steady state). More transactions should be simulated to bring the system in the steady state, but this will take weeks of computer time. Since this is not the purpose of our study, we chose not to do these time-intensive simulations. For the tables already show that our simple and fast analytic approximation is very useful. Moreover, in contrast to the influence of the long transactions on the mean response time g, their influence on the *tails* of G (and thus the error made by simulating only 4 million transactions) is negligible.

The Maximum Allowable Arrival Rate

We see from Table 1 and 2 that the performance from a transaction point of view, i.e., the mean and distribution of the generalized service time, depends heavily on the type of the execution time distribution. In the deterministic case, if we are willing to accept a mean generalized service time of 5 then from $g_D = e^{\lambda b}$ it follows that λb should be less than $\ln(5)$. In the exponential case $g_M = 1/(1-\lambda b)^2$, so λb should be less than $1 - (\sqrt{5}/5)$. For $b = 0.01$, this gives a maximum arrival rate λ of 161 and 55 in the deterministic and exponential case, respectively.

As for the *real-time* performance (which concerns the probability that the generalized service time is larger than some maximum allowable value t), we see that the system with exponential execution times has a much larger probability of missing deadlines than the system with deterministic execution times. If we have a (relative) deadline of 5 and we are willing to accept a miss probability of 0.05 (this is called $(5, 95)$-efficiency, see also Result 3.5), then for $b = 0.01$ in the deterministic case λ should be less than 79, while in the exponential case the bound for λ is 17.

In Table 3, for the deterministic and the exponential case, we show the maximum value λ^* of the arrival rate for various efficiency levels and for various values of b. Since the analysis only depends on λb and not on λ and b separately, the values of λ^* change proportionally to b.

b	$g = 5$		(t,α)-efficiency $(5,90)$		$(5,95)$		$(5,99)$	
	D	M	D	M	D	M	D	M
0.01	161	55	99	38	79	17	50	1.3
0.1	16	5.5	9.9	3.8	7.9	1.7	5.1	0.13
0.2	8.0	2.7	4.9	1.9	3.9	0.85	2.5	0.065

Table 3: *Maximum arrival rate λ^* for which g or (t,α)-efficiency is guaranteed*

A final remark about the maximum allowable value λ^* of the arrival intensity λ in relation to the number of servers. This value is 0.5 in an $M/M/\infty$ system with OCC if $b = 0.2$ and the requirement is that 96.5% of the transactions must meet deadline $t = 5$. Now consider a system without concurrency so with only 1 CPU. Let the execution time be exponentially distributed with mean 1. So in fact we are looking at an $M/M/1$ queue. The probability that the response time in an $M/M/1$ system is larger than 5 equals $e^{-(1-\lambda)5}$. Thus, the λ^* in the $M/M/1$ system such that

96.5% of the transactions meets deadline 5 is $1 + (\ln(0.035)/5) = 0.33$. Hence, going from 1 to ∞ CPUs increases the maximum allowable value of λ for which the system is still $(5, 96.5)$-efficient from 0.33 to 0.5, that is only by about 50%. For the same parameters, the $M/D/1$ queue compared to the $M/D/\infty$ queue with OCC for gives a λ^* of 0.66 compared to 3.5.

5 Broadcast OCC

The database system considered in Section 2 to 4 is governed by Pure OCC, that is, a transaction is aborted only at validation time (so after an execution run). In this section, we investigate the performance of a real-time database with Broadcast OCC.

Under Broadcast OCC, a transaction is aborted as soon as any data-item it has accessed is changed by another transaction. So no CPU time is wasted on completing unsuccessful execution runs. This implies that Broadcast OCC performs better than Pure OCC.

We analyze the mean and distribution of the generalized service time under Broadcast OCC in Section 5.1. Just as for Pure OCC, we use the assumption of complete data refreshment, i.e., in every run of a transaction all data to be used by the transaction are refreshed. In Section 5.2, we test our approximation of the generalized service time under Broadcast OCC against simulation.

Remark
Efficient implementations of both Pure and Broadcast OCC do not read anew all data-items of a transaction in every rerun. Only data-items that were involved in the conflict which caused the rerun are refreshed in the next run. We call this *minimal data refreshment*. If reading data is costly (i.e., takes a relatively large part of the execution time X), then the length of a rerun under OCC with complete data refreshment may be considerably larger than the length of a rerun under minimal data refreshment.
Under complete data refreshment, it is clear that Broadcast OCC outperforms Pure OCC. YU et al. [1993] stress that under minimal data refreshment, Pure OCC may perform better than Broadcast OCC. The reason for this is that reruns under Broadcast OCC may take longer than reruns under Pure OCC. Under Broadcast OCC, the earlier the first run is aborted, the fewer data-items have already been read, so the more items must be accessed in the next run; under Pure OCC, all items have already been read in the first run, so in a rerun only those data-items that caused conflicts must be reread. Therefore, under minimal data refreshment, the success probability of a rerun under Broadcast OCC may be smaller than that under Pure OCC. The gain that Broadcast OCC makes over Pure OCC by having a shorter first (unsuccessful) run may be undone by the larger number of reruns that is needed by Broadcast OCC.

In this paper, we only consider OCC algorithms with complete data refreshment. A study of OCC with minimal data refreshment is a topic for future research.

5.1 Analysis

As in Section 3 for Pure OCC, we make the following assumption.

Departure Assumption
The departure process of successfully committing transactions is a Poisson process with rate λ.

Let us consider a transaction that requires x time units of servicing. Under the Departure Assumption, a Poisson process with rate λ tries to invalidate the transaction during its execution. An invalidation attempt is successful with probability b. So successful invalidation attempts strike according to a Poisson process with rate λb. Thus the transaction is successful with probability $e^{-\lambda bx}$. If the first run of a transaction is successful, the mean generalized service time $g(x)$ equals x. Otherwise, if an invalidation attempt strikes the transaction at time t during the first run, the transaction is aborted and must be rerun. Then the mean generalized service time $g(x)$ equals $t + g(x)$. Hence, conditioning on the abort time t yields

$$g(x) = xe^{-\lambda bx} + \int_0^x (t + g(x))\lambda b e^{-\lambda bt}\,\mathrm{d}t.$$

Canceling and rearranging terms gives the following result.

Result 5.1 *Under Broadcast OCC, the mean generalized service time $g^{BC}(x)$ of a transaction of duration x satisfies*

$$g^{BC}(x) = (e^{\lambda bx} - 1)/\lambda b.$$

Conditioning on x leads to Result 5.2.

Result 5.2 *Under Broadcast OCC, the overall mean generalized service time g^{BC} satisfies*

$$g^{BC} = \frac{1}{\lambda b}\int_0^\infty (e^{\lambda bx} - 1)\,\mathrm{d}F(x)\,,$$

with F the execution time distribution.

If the execution time is deterministic with mean 1 then the overall mean generalized service time g_D^{BC} is given by

$$g_D^{BC} = (e^{\lambda b} - 1)/\lambda b.$$

If the execution time is exponentially distributed with mean 1 then the overall mean generalized service time g_M^{BC} is finite only if $\lambda b < 1$, in which case we have

$$g_M^{BC} = \frac{1}{1 - \lambda b}.$$

Note that, as expected, for all $x > 0$ the mean generalized service time $g^{BC}(x)$ of a transaction of duration x under Broadcast OCC is smaller than the mean generalized service time $g(x)$ under Pure OCC. Hence, we have

Result 5.3 *The overall mean generalized service time g^{BC} under Broadcast OCC with complete data refreshment is smaller than the overall mean generalized service time g under Pure OCC with complete data refreshment.*

We next derive the distribution function of the (overall) generalized service time under Broadcast OCC. Let us consider a transaction with execution time x. Denote the generalized service time of a transaction of duration x by G_x, and its distribution function by $G_x(t)$. G_x consists of a number of unsuccessful execution runs, all shorter than x, and one successful run, of length x. Denote by U_x the total length of the unsuccessful runs. Then G_x is distributed as $U_x + x$. The problem of deriving the distribution of U_x is essentially equivalent to the following basic problem in the theory of Poisson processes:

In a Poisson process with rate λb, what is the distribution of the total time U_x until the first moment at which an interarrival time starts that is larger than x?

Solving this problem is not trivial and even quite elaborate. In the Appendix, we give a derivation of the distribution function of U_x. Using the Appendix, the following results are immediate.

Result 5.4 *Under Broadcast OCC, the distribution function $G_x^{BC}(t)$ of the generalized service time of a transaction of duration x satisfies*

$$G_x^{BC}(t) = \sum_{n=0}^{\lfloor \frac{t}{x} \rfloor - 1} \frac{(-1)^n (\lambda b)^n}{(n+1)!} e^{-(n+1)\lambda b x} (t-(n+1)x)^n \left[n+1+\lambda b(t-(n+1)x)\right]$$

for $x \leq t$.

Result 5.5 *Under Broadcast OCC, the distribution function $G^{BC}(t)$ of the overall generalized service time G^{BC} satisfies*

$$G^{BC}(t) = \int_0^t G_x^{BC}(t)\, \mathrm{d}F(x)\,,$$

with F the execution time distribution.

5.2 Numerical Results

Again we consider the two cases, deterministic and exponential execution time, this time for Broadcast OCC. Results produced by simulations are compared with the approximative analysis.

In Table 4 we show L, g, $P(G > 1)$, $P(G > 2)$, and $P(G > 5)$ (dropping the superscript BC) for a system with deterministic transaction lengths. The same is shown in Table 5 for the exponential case. As with Pure OCC, the analytic values for $P(G > t)$ in the exponential case were computed by numerical integration. This time, all simulations were run until 1 million transactions committed.

A simulation study of the departure process under Broadcast OCC showed that the Departure Assumption is excellent for $b = 0.01$, and less accurate but still useful for $b = 0.1$ and 0.2, just as under Pure OCC.

For Broadcast OCC, analysis and simulation agree very well, both in the deterministic and in the exponential case.

		L_D		g_D		$P(G_D > 1)$		$P(G_D > 2)$		$P(G_D > 5)$	
λ	b	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim
20.0	0.01	22.14	22.14	1.11	1.11	0.18	0.18	0.018	0.018	0.0000	0.0000
40.0	0.01	49.2	49.2	1.23	1.23	0.33	0.33	0.062	0.062	0.0002	0.0002
80.0	0.01	122.6	122.7	1.53	1.53	0.55	0.55	0.19	0.19	0.0050	0.0051
2.0	0.10	2.21	2.22	1.11	1.11	0.18	0.18	0.018	0.017	0.0000	0.0000
4.0	0.10	4.92	4.94	1.23	1.23	0.33	0.33	0.062	0.063	0.0002	0.0002
8.0	0.10	12.26	12.41	1.53	1.55	0.55	0.56	0.19	0.20	0.0050	0.0059
1.0	0.20	1.11	1.11	1.11	1.11	0.18	0.19	0.018	0.018	0.0000	0.0000
2.0	0.20	2.46	2.48	1.23	1.24	0.33	0.34	0.062	0.064	0.0002	0.0003
4.0	0.20	6.13	6.29	1.53	1.57	0.55	0.57	0.19	0.20	0.0050	0.0077

Table 4: *Analysis versus simulation for Broadcast OCC with deterministic execution times*

In the deterministic case, the only significant errors occur in the less-than-1% tails for $b = 0.1$ and $b = 0.2$ when λ is large. As under Pure OCC, these errors are explained by the fact that for $b = 0.1$ and 0.2 the analysis does not recognize that the commit process (N_x) is more regular than Poisson, leading to an overestimation of the success probability and thus to an underestimation of the tail probabilities of G_D.
Although it is not shown in Table 5, the same error occurs in the tail of G for $b = 0.1$ and 0.2 in the exponential case. Further, in Table 5 some simulation results are printed in italics to indicate that a simulation of 1 million committed transactions was not sufficient to bring the system in the steady state (cf. the discussion in Section 4).

The system with deterministic execution times has considerably smaller generalized service times than the system with exponential execution times. However it is worth noting that the difference between g_D and g_M is much smaller than it was under Pure OCC.

Finally, we note that the tables clearly show that Broadcast OCC outperforms Pure OCC.

6 Conclusions

We analyzed the generalized service-time distribution in an $M/G/\infty$ queue with two variants of Optimistic Concurrency Control (OCC): Pure OCC and Broadcast OCC. Under Pure OCC, transactions are checked for data conflicts only at the end of their service. Under Broadcast OCC, a transaction is aborted and rerun as soon as any data-item it has accessed is changed by another transaction. Broadcast OCC obviously performs better than Pure OCC under the assumption of complete data refreshment (i.e. when a transaction is restarted, all data-items it needs are read anew from main memory). A comparison of the approximative analysis with simulation showed that the analysis is very accurate.

Although in practice the number of servers is never unlimited, this study is very useful to judge

		L_M		g_M		$P(G_M > 1)$		$P(G_M > 2)$		$P(G_M > 5)$	
λ	b	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim	Ana	Sim
20.0	0.01	25.0	25.0	1.25	1.25	0.39	0.39	0.17	0.17	0.032	0.032
40.0	0.01	67	67	1.67	1.66	0.41	0.41	0.21	0.21	0.060	0.060
80.0	0.01	400	*331*	5.0	*4.1*	0.44	0.44	0.27	0.27	0.12	0.12
2.0	0.10	2.50	2.50	1.25	1.25	0.39	0.39	0.17	0.17	0.032	0.032
4.0	0.10	6.7	6.7	1.67	1.68	0.41	0.41	0.21	0.21	0.060	0.061
8.0	0.10	40	*38*	5.0	*4.7*	0.44	0.44	0.27	0.27	0.12	0.12
1.0	0.20	1.25	1.25	1.25	1.25	0.39	0.39	0.17	0.17	0.032	0.033
2.0	0.20	3.33	3.40	1.67	1.70	0.41	0.41	0.21	0.21	0.060	0.061
4.0	0.20	20	*21*	5.0	*5.2*	0.44	0.45	0.27	0.27	0.12	0.12

Table 5: *Analysis versus simulation for Broadcast OCC with exponential execution times*

the value of having a large number of servers in OCC systems. At least two important conclusions can be drawn from this study.

The first conclusion has to do with the real-time performance of OCC systems. Even with an unlimited number of CPUs (servers), the real-time performance in terms of the mean response time and the distribution of the response time may be very low. Thus, in OCC systems adding servers is not the answer to all problems.

The second conclusion concerns the tremendous influence of long transactions on the (average) performance of OCC systems. For exponential execution times, and more general the case of moderately to highly variable execution times, OCC is not the appropriate concurrency control algorithm. Long execution times may lead to ridiculously long generalized service times and a serious waste of CPU power. As a consequence, it takes an extremely large number of transactions to bring a simulation of the system in the steady state.

Appendix

In this appendix, we derive the distribution of the total length U_x of a transaction's unsuccessful runs under Broadcast OCC with complete data refreshment. We first give three alternative ways of looking at this problem.

Consider a pedestrian who wants to traverse a one-way street. The pedestrian needs x time units to cross the street and can only cross if no cars are passing. If cars pass according to a Poisson process with rate λ, what is the distribution of the time until the pedestrian can cross the street?

This problem relates to the basic properties of a Poisson process. It can therefore also be formulated as follows.

In a Poisson process, what is the distribution of the total time until the first moment at which an interarrival time starts that is larger than x?

Or, more formally,

Consider a Poisson process with intensity λ. Denote the interarrival times by $\{X_1, X_2, \dots\}$. Let

$$N = \min\{n \mid X_i < x, i = 1, \dots, n,\ X_{n+1} \geq x\} \quad \textit{and} \quad U_x = \sum_{i=1}^{N} X_i.$$

Derive the distribution function of U_x.

In this appendix we solve the problem of the pedestrian. Let $u_x(t)$ be the density, and $U_x(t)$ be the distribution function of U_x.

Laplace Transform of U_x

Condition on the epoch at which the first arrival occurs. Denote by $H(y)$ the distribution function of X_1. Then

$$\begin{aligned} E\left[e^{-sU_x}\right] &= \int_0^\infty E\left[e^{-sU_x} \mid X_1 = y\right] \mathrm{d}H(y) \\ &= \int_0^x E\left[e^{-s(y+U_x)}\right] \mathrm{d}H(y) + \int_x^\infty e^{-s\,0}\, \mathrm{d}H(y). \end{aligned}$$

Thus,

$$\begin{aligned} E\left[e^{-sU_x}\right] &= E\left[e^{-sU_x}\right] \int_0^x e^{-sy}\, \mathrm{d}H(y) + e^{-\lambda x} \\ &= E\left[e^{-sU_x}\right] \frac{\lambda}{\lambda + s}(1 - e^{-(\lambda+s)x}) + e^{-\lambda x}, \end{aligned}$$

which yields

$$E\left[e^{-sU_x}\right] = \frac{e^{-\lambda x}}{1 - \frac{\lambda}{\lambda+s}(1 - e^{-(\lambda+s)x})} = \frac{(\lambda + s)e^{-\lambda x}}{s + \lambda e^{-(\lambda+s)x}},$$

or

$$\boxed{E\left[e^{-sU_x}\right] = e^{-\lambda x}\left(1 + \frac{\lambda(1 - e^{-(\lambda+s)x})}{s + \lambda e^{-(\lambda+s)x}}\right).}$$

Direct inversion of this transform to get $u_x(t)$ and/or $U_x(t)$ is not trivial. Therefore, we derive $u_x(t)$ and $U_x(t)$ in another way.

Density $u_x(t)$ of U_x

First we remark that

$$u_x(t) = \lim_{\Delta t \to 0} \frac{P(U_x \in (t, t + \Delta t))}{\Delta t}, \quad t \geq 0.$$

It is readily seen that

$$P(U_x \in (t,t+\Delta t)) = \sum_{n=1}^{\infty} P(X_1+\cdots+X_n \in (t,t+\Delta t),\ X_1<x,\ldots,X_n<x,\ X_{n+1}>x)$$
$$= e^{-\lambda x}\sum_{n=1}^{\infty} \int\cdots\int_{\substack{0<x_i<x,\ 1\le i\le n\\ x_1+\cdots+x_n\in(t,t+\Delta t)}} \lambda^n e^{-\lambda(x_1+\cdots+x_n)}\,\mathrm{d}x_1\cdots\mathrm{d}x_n.$$

So

$$u_x(t) = e^{-\lambda x}e^{-\lambda t}\sum_{n=1}^{\infty}(\lambda x)^n \int\cdots\int_{\substack{0<x_i<x,\ 1\le i\le n\\ x_1+\cdots+x_n=t}} \frac{1}{x^n}\,\mathrm{d}x_1\cdots\mathrm{d}x_n,$$

where the multiple integral is the density in t of the sum of n uniform$(0,x)$-distributed random variables. Using the notation $(y)_+ = \max\{y,0\}$, this density equals

$$\frac{1}{x^n(n-1)!}\sum_{\nu=0}^{n}(-1)^\nu\binom{n}{\nu}(t-\nu x)_+^{n-1},$$

see e.g. FELLER [1966], p. 27. Hence, for $t \ge 0$,

$$u_x(t) = e^{-\lambda x}e^{-\lambda t}\sum_{n=1}^{\infty}\frac{\lambda^n}{(n-1)!}\sum_{\nu=0}^{n}(-1)^\nu\binom{n}{\nu}(t-\nu x)_+^{n-1}.$$

Since the sum with infinitely many terms is not desirable for computational purposes, we simplify this expression further. We separate the term with $\nu = 0$ and interchange the order of summation:

$$u_x(t) = e^{-\lambda x}e^{-\lambda t}\sum_{n=1}^{\infty}\frac{(\lambda t)^{n-1}}{(n-1)!}\lambda + e^{-\lambda x}e^{-\lambda t}\sum_{\nu=1}^{\infty}(-1)^\nu\sum_{n=\nu}^{\infty}\lambda\frac{(\lambda(t-\nu x)_+)^{n-1}}{(n-1)!}\binom{n}{\nu}$$
$$= \lambda e^{-\lambda x} + \lambda e^{-\lambda x}e^{-\lambda t}\sum_{\nu=1}^{\infty}(-1)^\nu\left\{\frac{(\lambda(t-\nu x)_+)^\nu}{\nu!}+\frac{(\lambda(t-\nu x)_+)^{\nu-1}}{(\nu-1)!}\right\}e^{\lambda(t-\nu x)_+}$$

because

$$\sum_{n=\nu}^{\infty}\frac{z^{n-1}}{(n-1)!}\binom{n}{\nu} = \frac{z^\nu}{\nu!}e^z + \frac{z^{\nu-1}}{(\nu-1)!}e^z.$$

Since $(t-\nu x)_+ = 0$ for $\nu \ge \lfloor\frac{t}{x}\rfloor + 1$, this reduces to

$$u_x(t) = \lambda e^{-\lambda x} + \lambda e^{-\lambda x}e^{-\lambda t}\sum_{\nu=1}^{\lfloor\frac{t}{x}\rfloor}(-1)^\nu\left\{\frac{(\lambda(t-\nu x))^\nu}{\nu!}+\frac{(\lambda(t-\nu x))^{\nu-1}}{(\nu-1)!}\right\}e^{\lambda(t-\nu x)},$$

or

$$\boxed{u_x(t) = \lambda e^{-\lambda x} + \sum_{\nu=1}^{\lfloor\frac{t}{x}\rfloor}\frac{(-1)^\nu\lambda^\nu}{\nu!}e^{-(\nu+1)\lambda x}\left[\nu+\lambda(t-\nu x)\right](t-\nu x)^{\nu-1}.}$$

Distribution Function $U_x(t)$ of U_x

First note, that U_x has probability mass $e^{-\lambda x}$ in 0, so $U_x(0) = e^{-\lambda x}$.

$$U_x(t) = P(U_x \leq t) = U_x(0) + \int_0^t u_x(y)\,\mathrm{d}y$$

Using again that the term $(-\lambda)^n e^{-(n+1)\lambda x}[n + \lambda(y - nx)](y - nx)^{n-1}/n!$ occurs in $u_x(y)$ for all $y \geq nx$ with $n \geq 1$, we get

$$\begin{aligned} U_x(t) &= e^{-\lambda x} + \lambda t e^{-\lambda x} + \sum_{n=1}^{\lfloor \frac{t}{x} \rfloor} \int_{nx}^{t} \frac{(-1)^n \lambda^n}{n!} e^{-(n+1)\lambda x} \left[n + \lambda(y - nx)\right](y - nx)^{n-1}\,\mathrm{d}y \\ &= e^{-\lambda x} + \lambda t e^{-\lambda x} + \sum_{n=1}^{\lfloor \frac{t}{x} \rfloor} \frac{(-1)^n \lambda^n}{n!} e^{-(n+1)\lambda x} \int_{nx}^{t} \left[n + \lambda(y - nx)\right](y - nx)^{n-1}\,\mathrm{d}y, \end{aligned}$$

so for $t \geq 0$,

$$\boxed{U_x(t) = \sum_{n=0}^{\lfloor \frac{t}{x} \rfloor} \frac{(-1)^n \lambda^n}{(n+1)!} e^{-(n+1)\lambda x}(t - nx)^n \left[n + 1 + \lambda(t - nx)\right].}$$

The check $\dfrac{\mathrm{d}U_x(t)}{\mathrm{d}t}$ indeed yields $u_x(t)$ for $nx \leq t < (n+1)x$ with $n \geq 0$.

Acknowledgment

We thank Jacques Resing for deriving the distribution function of the total length of a transaction's unsuccessful runs under Broadcast OCC (see Appendix).

References

DOOB, J.L. [1953]. *Stochastic processes.* John Wiley & Sons, New York.

FELLER, W. [1966]. *An introduction to probability theory and its applications, vol. II.* John Wiley & Sons, Inc., New York.

KENDALL, D.G. [1964]. Some recent work and further problems in the theory of queues. *Theory of Probability and its Applications*, **9**, 1–13.

NEWELL, G.F. [1966]. The $M/G/\infty$ queue. *SIAM J. Appl. Math.*, **14**, 86–88.

SASSEN, S.A.E, AND J. VAN DER WAL [1997a]. The response-time distribution in a real-time database with optimistic concurrency control and exponential execution times. In V. Ramaswami and P.E. Wirth (editors), *Proceedings of the 15th International Teletraffic Congress - ITC 15, Washington, DC, USA, 22–27 June, 1997*, pages 145–156. North-Holland.

SASSEN, S.A.E., AND J. VAN DER WAL [1997b]. The response-time distribution in a real-time database with optimistic concurrency control and constant execution times. Technical Report COSOR 97-07, Dept. of Mathematics and Computing Science, Eindhoven University of Technology.

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