Performance analysis for real-time databases

Performance Analysis for Real-time Databases

Simone A.E. Sassen
Statistics and Operations Research Group
Dept. of Mathematics and Computing Science
Eindhoven University of Technology

February 23, 1995

Abstract

In this talk we focus on uncertainty in real-time databases. In most applications of real-time databases, no exact information is available about the arrival times, the sizes and the deadlines of future transactions. Due to this uncertainty it is not sufficient to measure the performance of a real-time database in terms of the *average* response time of a transaction; one has to obtain an approximation for its *distribution*. The ultimate result should be to find an approximation for the probability that a transaction meets its deadline.

Performance Requirements

As has been explained in the talk of Maarten P. Bodlaender, real-time databases have to satisfy both database and real-time requirements.
Database requirement:

- execution of transactions must preserve consistency.

Real-time requirements:

- some transactions can only be executed in a certain time interval
- transactions must meet their deadlines.

The purpose of the performance analysis for real-time databases (RTDB's) is to look at schedulers that preserve the consistency of the database and to investigate how well the real-time requirements are satisfied.

Performance questions that could be of interest are: (1) which percentage of the transactions meets its deadline, (2) what is the transaction throughput of the RTDB, (3) how reliable is the RTDB (how small is the probability of deadlock), and so on. Motives for an interest in question (1) are

- to offer a high customer service level
  (Rabobank, PTT Telecom, Ericsson)→ **90% ?**

- it is very important for the control of (physical) processes, where deadlines may be quite firm
  (ECT, research project at INRIA) → **99%** or even **99.99%** ?

To be able to give an answer to performance questions it is necessary to investigate the **response time** of a (type of) transaction in the system. Having a good approximation for the average response time and the standard deviation of the response time could already enable us to judge if the RTDB can meet a performance level of about 90%. However when one wants to distinguish a level of 99% from 99.9%, a very accurate approximation of the probability distribution of the response time is needed. It is not very likely that we will be able to find such an accurate approximation for the response time distribution, but the aim is to at least derive approximations that are good enough for RTDB design where performance levels of about 90% or 95% are required.

How to Model?

Uncertain factors that influence the response time distribution are:

1. **The arrival instants of transactions.**
   In a real-time environment it is usually not known beforehand when arrivals of transactions will take place. There may be some type of transaction that comes within regular (known) intervals; other transactions arrive irregularly, which has to be modelled as a stochastic process.

2. **Sizes of transactions** (# data items needed + amount of work).
   These may also vary per transaction (type). The more data items are needed, the more data contention can occur so the longer the response time of the transaction might be. The amount of work a transaction requires can vary since this involves e.g. communication times and computing times. A probability distribution for the size and the amount of work of the transactions is needed.

3. **Deadlines and priorities of transactions.**
   Jobs with high priority will increase the response time of lower priority jobs.

4. **Arrival locations of transactions** (in distributed databases).
   Which percentage of the transactions arrives at which location?

5. **Data-item popularity.**
   If there are some popular data-items that are used by a lot of transactions and (for example) 2-phase locking is used to protect the data, the response times of these transactions can grow quite large since these transactions must wait for the popular data-items to become available. If there are no popular data-items (uniform data-access) the influence of data contention on the response times might be very small.

6. **Transactions can trigger new transactions.**
   This would mean that the times between arrivals are not independent.

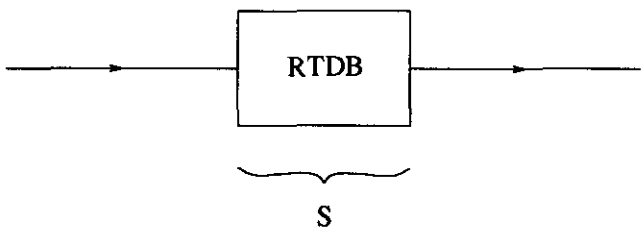


Figure 1: THE RTDB AS A BLACK BOX

How the above factors should be modelled depends on the specific application. To be able to formulate a good model for deriving an approximation for the distribution of the response time, it is vital to know which of the above elements play an important role in the various applications of the user group, and which elements can be ignored. For instance: is the data-access uniform, are the transactions of about the same size and type, what is the conflict probability between two transactions, and do transactions arrive regularly?
Hopefully this workshop and further discussions with the members of the user committee can contribute to getting a clear view of what the important elements are for each of the applications of the user committee.

Performance Analysis by Use of Queueing Models

A way of deriving approximations for the response time of transactions is by stochastic modelling. If we see the response time as a stochastic variable that is influenced by the elements mentioned in the previous paragraph, we can use queueing theory for approximating the response time distribution.

Queueing models have their origin in the study of design problems of automatic telephone exchanges and were first analysed by the queueing pioneer A.K. Erlang in the early 1900s. In the last 30 years, quite some progress has been made in the theory of queueing models. They have been applied e.g. to the design of computer systems, telecommunication networks and many problems in manufacturing. Our feeling is that they can be very useful for evaluating the performance of real-time databases.

The most abstract way of seeing a real-time database system is as in Figure 1. Transactions are entering the system, and after having spent a time S in the system they leave. What happens inside the 'black box' is not clear and will depend on the application. The response time S may consist of both **waiting time** (on locks or hardware) and **service time** (for the actual execution). For a good approximation of the response time a more detailed view of the database system (the black box) is needed. How many processors are available at which

sites? Where are the data-items located? Which locking protocol is used? Modelling the database system as a queueing system requires information on when and where in the system transactions have to wait, and which service discipline is used.

Suppose we had only one type of transaction with fixed transaction size, and we made the following assumptions:

- The transactions arrive independently of each other.
- The time between 2 arrivals is exponentially distributed. In other words, the arrival process is a Poisson process.
  This assumption is often used in queueing models. Reasons are:
  1. it is an excellent approximation in the case of a large number of potential customers where each customer has a very small probability of arriving in a specific time interval (think of the 7 million people that have a telephone: each of them has a very small probability of making a phone call in a specific period of (say) 1 second);
  2. it has computationally attractive properties that simplify an analysis, e.g. the *memoryless* property.

  Note that the assumption of a Poisson arrival process may not be reasonable if there is a great regularity in the arrival of transactions.
- The service time of a transaction is exponentially distributed.
  This assumption is usually not a good approximation but because of the nice properties of the exponential distribution it is often taken to start with.

On the last page of this text, three representations of the real-time database system are given, based on the above assumptions.

I The top one is the most rigorous way of simplifying the database system and is easiest to analyse. The database can handle only one transaction at a time (the service discipline is FCFS), and all other transactions wait in a queue. For this model an exact expression exists for the distribution of the response time. When deadlines are taken into account by giving the transaction with the earliest deadline the highest priority, it is still possible to find (an approximation for) the distribution of the response time.

II The second figure releases the assumption of only one server. It assumes several parallel servers and thus can handle more transactions at a time. When more than k transactions are present, only k can be served and the remainder has to wait in a queue. For this queueing model an exact expression for the distribution of the response time is known. For generally distributed service times approximations for the response time distribution are available.
However when the locking protocol is taken into account, approximating the

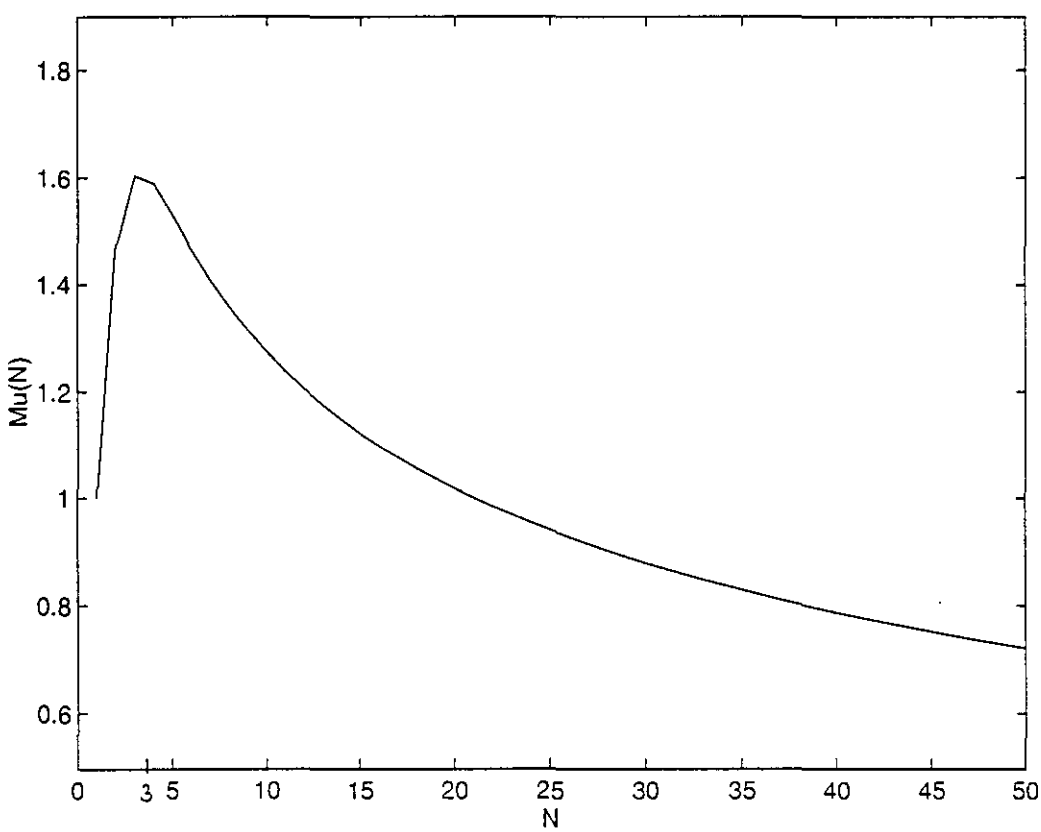
response time becomes more tedious. Because then a transaction that is executing at server 1 could have to wait for some data-item currently in use by the transaction executing at server k.

III The last figure shows a model for optimistic concurrency control that we have been investigating in more detail. It is assumed that each transaction demands some CPU-time (where the CPU is a single server that can handle only one transaction at a time, the rest has to wait in a queue), after which some computations have to be done at another site or processor (represented as an infinite server, so without capacity restrictions). After each visit to the CPU the transaction leaves the system with some probability or goes through another cycle with 1 minus that probability. A transaction that tries to leave enters a validation phase, in which it checks if a conflicting transaction committed during its execution. If no conflicting transaction committed during the execution of the validating transaction, the transaction can commit and leave the database system. Otherwise, the transaction goes back to the CPU and has to be rerun.

An approximate analysis of the system has resulted in the conclusion that the system is not ergodic, i.e. if the number of transactions allowed in the system is not restricted to some value N, there may be an infinite number of transactions cycling in the modelled system. Consequently, the performance of the system (the throughput of the system) decreases to zero as the number N of transactions allowed tends to infinity. For any choice of input values for the conflict probability and the speeds of the servers, it can be shown that this is the case.

The behaviour of the system is intuitively understandable, for the probability that a transaction has to be rerun depends on the number of transactions that has committed during its execution. Now, as the number of transactions in the system is larger, it is more likely that a conflicting transaction has committed during the execution of the transaction that tries to validate, so the higher is the probability that the validating transaction has to be rerun. This makes the system even fuller and eventually more transactions are entering the system per unit time than are leaving, resulting in a throughput that diminishes to zero.

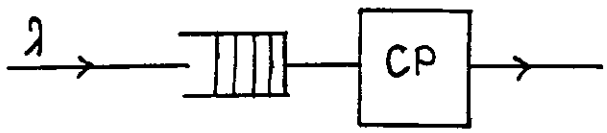
A typical illustration of the degrading performance is given in the plot below. Here we also see that there is a value of N for which the throughput of the system is maximal.



$\mu(N) \longrightarrow 0$ FOR $N \longrightarrow \infty$.

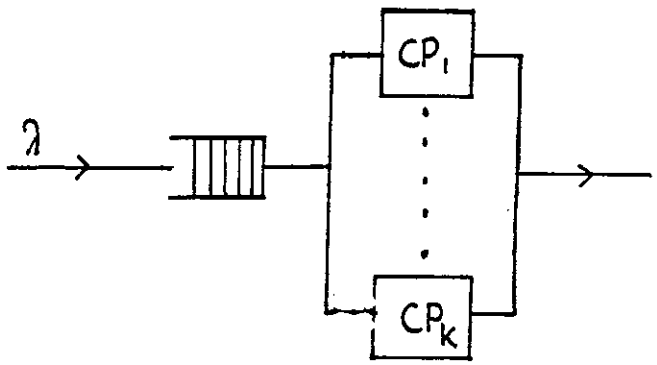
$\mu_1 = 2$
$p = 0.1$
$\lambda = 0.5$

I



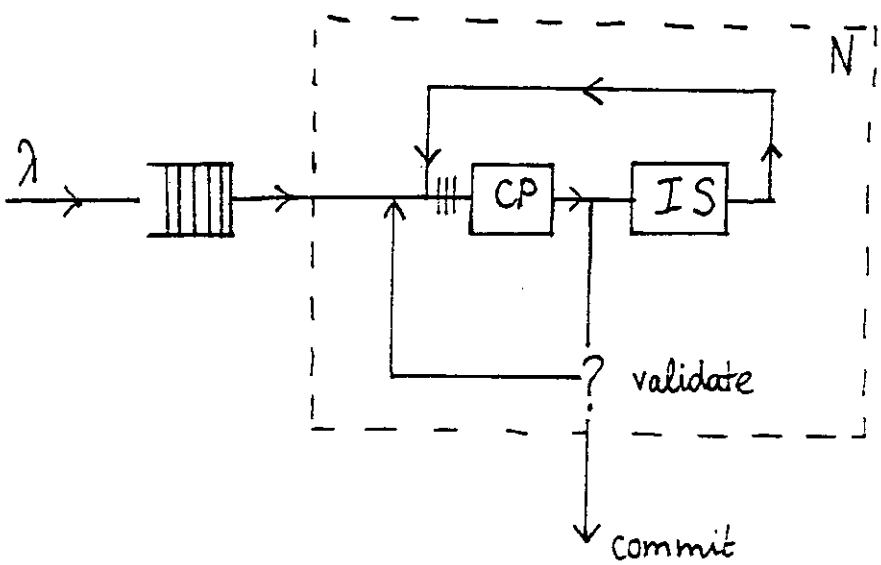
FCFS, 1 server

II



k parallel servers

III



- fixed transaction size
- p = Probability that 2 transactions conflict