Dependability and Performance Measures for the Database Practitioner

Toby J. Teorey and Wee Teck Ng

Abstract -- We estimate the availability, reliability, and mean transaction time (response time) for repairable database configurations, centralized or distributed, in which each service component is continuously available for repair. Reliability, the probability that the entire transaction can execute properly without failure, is computed as a function of mean time to failure (MTTF) and mean time to repair (MTTR). Mean transaction time in the system is a function of the mean service delay time for the transaction over all components, plus restart delays due to component failures, plus queuing delays for contention. These estimates are potentially applicable to more generalized distributed systems.

***Index terms* -- Database performance estimation, response time, reliability, dependability, restart delays, queuing delays, mean time to failure.**

I. Introduction

The increasing availability and importance of centralized, distributed, and multidatabases raises serious concerns about their dependability in a fragile network environment, much more than with centralized databases. Although the major impetus for distributed data is to increase data availability, it is not always clear whether the dependability of the many hardware and software components of a distributed system is such that the level of availability desired is actually provided. Performance of a database system is closely related to dependability, and it cannot be good if the dependability is low.

Failures occur in many parts of a computer system: at the computer sites, the storage media (disk), communication media, and in the database transactions, [1], [3], [6], [11]. Site failures may be due to hardware (CPU, memory, power failure) or software system problems. Disk failures may occur from operating system software bugs, controller problems, or head crashes. In the network, there may be errors in messages, including lost messages and improperly ordered messages, and line failures. Transaction failures may be due to bad data, constraint failure, or deadlock [7]. Each of these types of failures contributes to the degradation of overall dependability of a system.

A significant amount of research has been reported on the subject of dependability of computer systems, and a large number of analytical models exist to predict reliability for such systems [8], [13], [14]. While these models provide an excellent theoretical foundation for computing dependability, there is still a need to transform the theory to practice [2], [12], [16]. Our goal is to provide the initial step in such a transformation with a realistic set of system parameters, a simple analytical model, and comparison of the model predictions with a discrete event simulation tool.

Based on the definitions in [8], [15] dependability of any system can be thought of as composed of three basic characteristics: availability, reliability and serviceability. The *steady-state availability* is the probability that a system will be operational at any random point of time, and is expressed as the expected fraction of time a system is operational during the period it is required to be operational. The *reliability* is the probability that a system will perform its intended function properly without failure and satisfy specified performance requirements during a given time interval [0,t] when used in the manner intended. The *serviceability* or *maintainability* is the probability of successfully performing and completing a corrective maintenance action within a prescribed period of time with the proper maintenance support.

We look at the issues of availability and reliability in the context of simple database transactions (and their sub transactions) in a network environment where the steady-state availability is known for individual system components: computers, networks, the various network interconnection devices, and possibly their respective sub components. A transaction path is considered to be a sequential series of resource acquisitions and executions, with alternate parallel paths allowable. We assume that all individual system components, software and hardware, are repairable [8]. A *non repairable* distributed database is one in which transactions can be lost and the system is not available for repair. In a *repairable* distributed database all components are assumed to be continuously available for repair, and any aborted transaction is allowed to restart from its point of origin. We will only consider repairable databases here.

Serviceability is assumed to be deterministic in our model, but the model could be extended for probabilities less than 1 that the service will be successfully completed on time.

II. Availability

Availability can be derived in terms of the mean time to failure (MTTF) and the mean time to repair (MTTR) for each component used in a transaction. Note that from [15] we have the basic relationship for mean time between failures (MTBF):

$$MTBF = MTTF + MTTR \qquad (1)$$

The steady state availability of a single component i can be computed by

$$A_i = MTTF_i/(MTTF_i + MTTR_i)$$

$$= MTTF_i/MTBF_i \qquad (2)$$

Let us look at the computation of steady state availability in the network underlying the distributed or multidatabase. In Fig. 1a two sites, S1 and S2, are linked with the network link L12. Let A_{S1}, A_{S2}, and A_{L12} be the steady state availabilities for components S1, S2, and L12, respectively.

Assuming that each system component is independent of all other components, the probability that path S1/L12/S2 is available at any randomly selected time t is the product of the individual availabilities in series:

$$A_{S1/L12/S2} = A_{S1}*A_{L12}*A_{S2} \qquad (3)$$

Extending the concept of availability to parallel paths as shown in Fig. 1b, we factor out the two components, S1 and S2, that are common to each path:

$$A_{S1//S2} = A_{S1}*A_{S2}*[\text{availability of the connecting paths between S1 and S2}] \qquad (4)$$

Eq. 2 states that the total path from site S1 to site S2 has three serial components: S1, S2, and the two possible connecting paths between the sites. We simply partition the whole path into three serial parts and apply Eq. 1 to them to determine the total path availability. Now we need to determine the actual value of the third component of availability, the connecting paths. This is determined by the well-known relationship for parallel independent events that states that the total availability of a parallel path is the sum of the serial availability of each of the two separate paths, minus the product of their serial availabilities.

$$A_{S1//S2} = A_{S1}*A_{S2}*[A_{L12} + A_{L13}*A_{S3}*A_{L32} - A_{L12}*A_{L13}*A_{S3}*A_{L32}] \qquad (5)$$

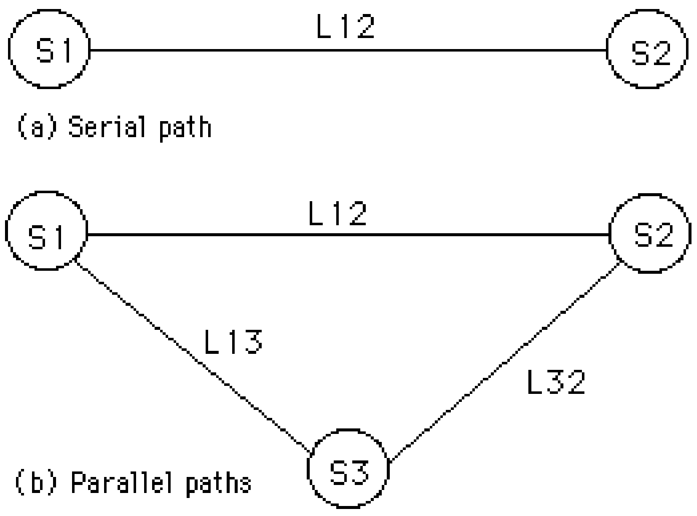

Figure 1. Network paths for a distributed database

We now have the basic relationships for serial and parallel paths for steady-state availability. We note that if query optimizers pick the shortest path without regard to availability, the system could reverse the decision if the selected path is not available. Because of the extreme complexity of computing reliability for parallel paths, we focus the remaining discussion on serial paths, only, to illustrate the basic concepts of combining reliability and performance into a single measure.

III. Reliability

An estimate of availability is limited to a single point in time. We now need to estimate the reliability for an entire transaction (including queries and/or updates), and in Sec. 4 the mean transaction completion time for a repairable distributed or multidatabase that has automatic restarts. Reliability is the probability that the entire transaction can execute properly (over a given time interval) without failure, and we need to compute the estimated mean reliability over a time duration [0,t], where t is the mean delay experienced over the system during the transaction execution.

For tractability we first assume that the number of failures of each system component is exponentially distributed:

$$P_i(k,t) = (mt)^k * e^{-mt}/k! \qquad (6)$$

This is the probability that there are exactly k failures of component i in time interval t, where m is the mean number of failures per unit time. The probability that there are no failures of component i in time interval t is:

$$P_i(0, t) = e^{-mt} \qquad (7)$$

Let $MTTF_i$ = mean time to failure on component i,

$MTTR_i$ = mean time to repair for component i, and

D = mean service delay time for the transaction.

We can now compute the reliability of the transaction at component i; for example the joint probability that the transaction has no failures while actively using component i, which is equal to the conditional probability that the component i is reliable over the interval (0, D) times the probability that the component i is available at the beginning of the same interval. That is,

$$P_i(0, D) = e^{-m_i*D} * A_i \qquad (8)$$

where m_i = failure rate = expected number of failures per unit time = $1/MTTF_i$. The reliability of the entire transaction is the product of the (assumed independent) reliabilities of the transaction for each component.

A. *Example: Database transaction reliability*
Let us now apply the relationship on Eq. 8 to the serial path database configuration over the network in Fig. 1a. The transaction reliability is equal to the probability that the transaction can be completed without failure from initiation at site S1, local access to the data in site S2, and returning with the result to site S1. We assume that the transaction is successful only if all components are active the entire time required to service the transaction. We are given the mean delay experienced by each sub transaction on each component resource, derived from known characteristics of the network and database system.

$P(0,D)$ = transaction reliability = $P_{S1}(0, D)*P_{L12}(0, D)*P_{S2}(0, D)$ (9)

Let
QIT = query initiation time(CPU) = 1 ms
PTT = packet transmission time = 8 ms (T1 link @ 1.544 Mb/s, packet size 1544 Bytes)
PD = packet propagation delay = 10 ms (assumed 2000 km distance, degraded electronic speed 200 km/ms)
QPT = query processing time(CPU & I/O) = 200 ms
n=number of packets in the result of the query = 5
QRDT = query result display time (CPU & I/O) = 60 ms
$MTTF_i$ = 10 hours (36,000 sec.) for each component i
$MTTR_i$ = .5 hour (1800 sec.) for each component i ($MTBF_i$ = 10.5 hours)
A_i = 10/(10.5) = .9524

Let us define the mean service delay time as the sum of all the non overlapped delays from site S1 to site S2 and returning the result to site S1:

$$\begin{aligned} D &= QIT + PTT + PD + QPT + n*PTT + PD + QRDT \\ &= 1 + 8 + 10 + 200 + 5*8 + 10 + 60 \text{ ms} \\ &= 329 \text{ ms } (.329 \text{ sec.}) \end{aligned}$$

Applying Eq. 9 we obtain the transaction reliability:

$$\begin{aligned} P(0,D) &= [e^{-.329/36000} (.9524)]^3 \\ &= .8639 \end{aligned}$$

IV. Mean Transaction Time

The mean transaction time in the system is a function of the mean service delay time for the transaction over all components, plus restart delays, plus queuing delays for contention as shown in Fig. 2. Once the mean service delay time (D) is known, we can then compute the mean completion time (C), taking into account the restart delays. Finally, we compute the mean transaction time in the system (T) from known queuing formulas.

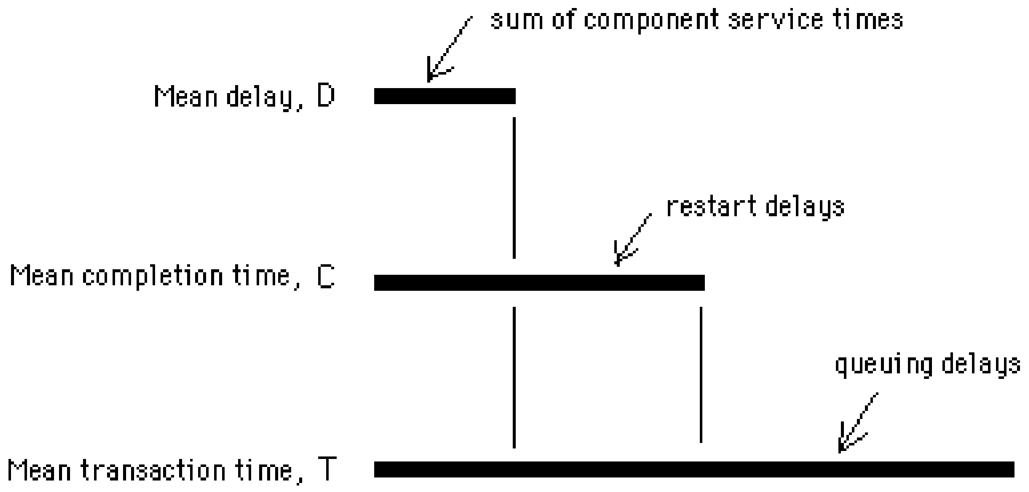

Figure 2. Components of mean transaction time

If there are no queuing delays, then we can easily estimate mean completion time, C, the combination of mean service and restart delays, by computing the probability of different completion times possible. For example, if D is the mean service delay time, C can be easily derived, assuming p, the reliability of a transaction, and for every failure, an average time to failure and recovery, D/2 + MTTR, where D/2 is the mean time to failure of the transaction, given random failures of a collection of components over which that transaction must successfully execute.

$$C = p*D + q*p*(D + D/2 + MTTR) + q^2*p*(D + 2*D/2 + 2*MTTR)$$

$$+ q^3*p*(D + 3*D/2 + 3*MTTR) + ...$$

$$= p*D + q*p*D + q^2*p*D + q^3*p*D + ... + q*p*(D/2 + MTTR)$$

$$+ 2q^2*p*(D/2 + MTTR) + 3q^3*p*(D/2 + MTTR) + ...$$

$$= p*D*(1 + q + q^2 + q^3 + ...) +q*p*(D/2 + MTTR)*(1 + 2q + 3q^2 + 4q^3 + ...)$$

$$= D + (q/p)*(D/2 + MTTR) \qquad (10)$$

by noting that (1 + q + q2 + q3 + ...) = 1/(1-q) = 1/p

and $(1 + 2q + 3q^2 + 4q^3 + ...) = 1/(1-q)^2 = 1/p^2$.

If, on the other hand, queuing delays do exist (e.g. in any shared resource system), the simple model of Eq. 10 breaks down, and mean transaction time, T, can be derived by noting the similarity between queues with breakdowns and preemptive priority queues [4]. We model our reliability problem with a preemptive priority queue. Consider a queuing system with two priority classes in which the high priority jobs can pre-empt any low priority job. This is equivalent to our reliability problem, where the transaction process corresponds to the low priority job (2) and the failure process corresponds to an arrival of a high priority job (1). The mean transaction time is thus the mean service time of the low priority job, and the exact solution [1] is:

$$T = \frac{(1/\mu_2)(1-\rho_1-\rho_2)+R_2}{(1-\rho_1-\rho_2)(1-\rho_1)} \qquad (11)$$

where R_2 is the mean residual time:

$$R_2 = \frac{\lambda_1 \overline{X_1^2} + \lambda_2 \overline{X_2^2}}{2} \qquad (12)$$

ρ_1 and ρ_2 are the utilization of the failure and transaction processes, respectively, μ_2 is the transaction service rate, and $\overline{X_i^2}$ is the second moment of service time. Eq. 11 is valid for any general service time.

Since each component is assumed to fail independently, the *k* independent Poisson failure processes on each component on the network can be combined into a single process with arrival rate equal to sum of the rates of each individual processes [1]. We assume that the transaction is successful only if all components are active the entire time required to service the transaction, so the tandem network in Fig. 1a can be simplified to a single server with mean transaction time given in Eq. 11.

A. *Example: Mean Transaction Time*

Let us now apply the relationship in Eq. 11 to the simple database configuration over the network in Fig. 1a. The transaction reliability is equal to the probability that the transaction can be completed without failure from initiation at site S1, local access to the data in site S2, and returning with the result to site S1. We are given the mean delay experienced by each sub transaction on each component resource, derived from known characteristics of the network and database system.

Given D =.329 sec, query arrival rate=2.5/sec, MTTR=0.5 hour, MTBF=10 hours, the configuration in Fig. 1a, and assuming that the transaction and failure processes follow the Poisson distribution, we derive:

	Failure Process	**Transaction Process**
Arrival Rate (/sec)	$\lambda_1 = \frac{3}{MTBF}$ =7.937 x 10^{-5}	λ_2=1.5
Service Rate (/sec)	$\mu_1 = \frac{1}{MTTR}$ =5.556 x 10^{-4}	$\mu_2 = \frac{1}{MD}$ =3.040
Utilization	ρ_1=0.1429	ρ_2=0.4935
2nd moment of Service Time (sec^2)	$\overline{X_1^2} = \frac{2}{\mu_1^2}$ =6.480 x 10^6	$\overline{X_2^2} = \frac{2}{\mu_2^2}$ =0.2165
Mean Residual Time (sec)	-	$R_2 = \frac{\lambda_1 \overline{X_1^2} + \lambda_2 \overline{X_2^2}}{2}$ =257.305
Mean Transaction Time (sec)	-	$T = \frac{(1/\mu_2)(1-\rho_1-\rho_2)+R_2}{(1-\rho_1-\rho_2)(1-\rho_1)}$ =825.89

V. Simulation Model

An event-driven discrete simulation model was implemented to verify the analytical model, with both models using the same service time assumptions. The simulation program was written in C and SMPL simulation routines [5], [9], [10]. The only modification made to SMPL was to incorporate a 48-bit random number generator with better spectral properties.

The network configuration for the distributed database is depicted in Fig. 1a. It consists of two sites, S1 and S2, linked by the network link L12. The main objective of the performance analysis is to evaluate the mean transaction time between S1 and S2 under unreliable network conditions. The parameters used in this study are tabulated below.

Simulation Parameters	**Values**
Transaction arrival rates	range: 0 to 2.85 per sec
Aggregate transaction service time*	.329 sec
Mean time to failure (MTTF)	300 sec
Mean time to repair (MTTR)	5 sec

*Note: The transaction service time is the sum of all delays incurred in two nodes and a link.

The results were analyzed using the batch mean approach with 20,000 transactions [9], and have an accuracy of 10 percent (or better) at a 99 percent confidence level. The analytical model and simulation results for mean transaction time, T, are shown in Fig. 3. The analytical results match the simulation standard at all arrival rates within 5 percent.

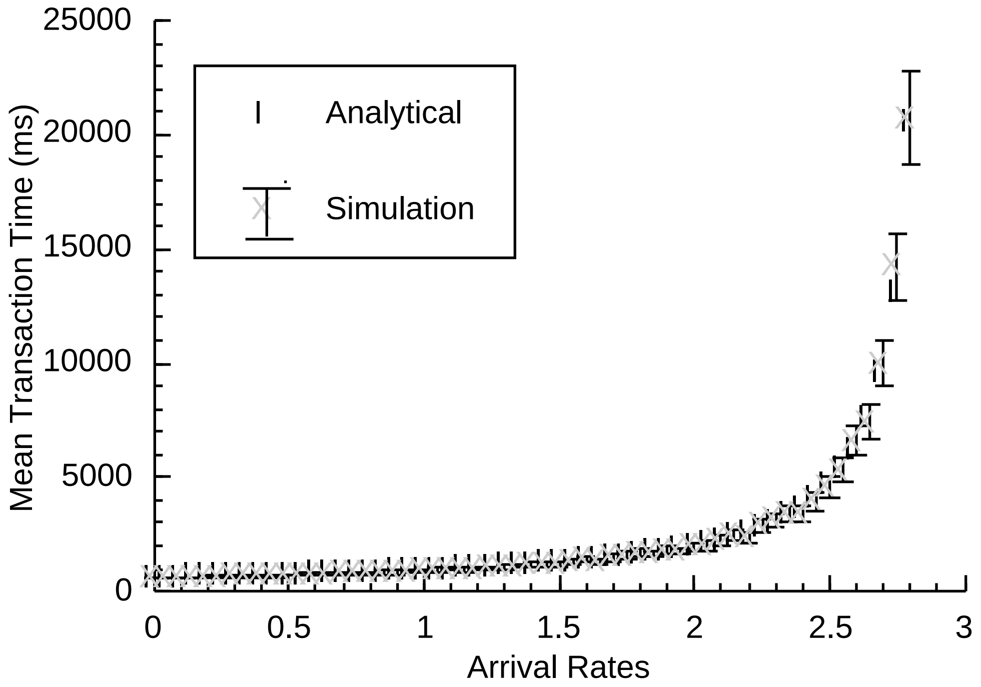

Figure 3. Mean Transaction Time (Repairable Network)

VI. Conclusions

We have derived expressions for availability and reliability in a repairable distributed database system for serial transactions. Reliability is derived in terms of the probability of success of an entire transaction over many system components. The mean transaction time in the system is computed as a function of restarts due to component failure and to traffic congestion. The mean transaction time model compares favorably with discrete simulation results from a distributed database configuration. Future work will involve full validation with live systems using the system parameters defined in our model, which should be straightforward to set up and monitor.

REFERENCES

[1] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice Hall, 1992.

[2] S. Ceri and G. Pelagatti, *Distributed Databases: Principles and Systems.* New York, NY: McGraw-Hill, 1984.

[3] J. B. Dugan, "On measurement and modeling of computer systems dependability: A dialog among experts," *IEEE Transactions on Reliability*, vol. 39, no. 4, pp. 506-510, Oct. 1990.

[4] A. Federgruen and L. Green, "Queueing systems with service interruptions," *Operations Research*, vol. 34, no. 5, pp. 752-768, Sept.-Oct. 1986.

[5] P. Fishwick, "SIMPACK: Getting started with simulation programming in C and C++," University of Florida Tech. Report TR 92-022, 1992.

[6] J. Gray, "A census of tandem system availability between 1985 and 1990," *IEEE Transactions on Reliability,* vol. 39, no. 4, pp. 409-418, Oct. 1990.

[7] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann, 1993.

[8] A. M. Johnson, Jr. and M. Malek, "Survey of software tools for evaluating reliability, availability, and serviceability," *ACM Computing Surveys*, vol. 20, no. 4, pp. 227-269, Dec. 1988.

[9] A. Law and W. Kelton, *Simulation Modeling and Analysis.* New York, NY: McGraw-Hill, 1991.

[10] M. MacDougall, *Simulating Computer Systems: Techniques and Tools*. Cambridge, MA: MIT Press, 1987.

[11] R. A. Maxion, and F. E. Feather, "A case study of ethernet anomalies in a distributed computing environment," *IEEE Transactions on Reliability*, vol. 39, no. 4, pp. 433-443, Oct. 1990.

[12] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[13] W. H. Sanders, and J. F. Meyer, "METASAN, A performability evaluation tool based on stochastic activity networks," *Proc. 1986 Fall Joint Computer Conference*, AFIPS, New York, 1986, pp. 807-816.

[14] R. A. Sahner and K. S. Trivedi, "Reliability modeling using SHARPE," *IEEE Transactions on Reliability,* vol. 36, no. 2, pp. 186-193, June 1987.

[15] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*. Bedford, MA: Digital Press, 1982.

[16] T. J. Teorey, *Database Modeling and Design: The Fundamental Principles*, 2nd Edition. San Francisco, CA: Morgan Kaufmann, 1994.