the Availability Digest

www.availabilitydigest.com

Calculating RPO

March 2010

Specifying RPO

In a redundant data-processing system, RPO, the Recovery Point Objective, is the amount of data loss that is acceptable following a node failure. The specification of RPO becomes especially important for critical applications using data replication between active/active nodes or from an active node to a backup node.

Tighter RPOs mean faster servers and greater bandwidth between the nodes. Therefore, a real cost is associated with the RPO specification. As a consequence, it is important to be able to reasonably estimate the RPO that a given system can meet. If it is insufficient, additional expense may be needed to enhance the system. There is a tradeoff between the cost of lost data and the system and communication costs to protect that data that must be considered by management.

Of course, it is unrealistic to think that there is an absolute limit to data loss. Some obscure event may result in a significant loss of data even though the expected loss might be quite small. Therefore, the RPO should be expressed as a probability that no more than a certain amount of data will be lost. For instance, the RPO may state that

> "99% of node failures will result in no more than 300 milliseconds of lost data at 100 transactions per second."

Since there will be on the average 30 transactions executed in 300 msec. at 100 transactions per second, this could equally as well be expressed as

> "99% of node failures will result in no more than 30 lost transactions at 100 transactions per second."

We will deal in this article with the first form of the RPO expressed in time rather than in transactions.

If an SLA (service level agreement) specifies a fixed RPO, an accurate estimate will indicate the probability that the SLA will be violated (a useful metric to a service provider to let it properly price an SLA). In this article, we discuss a method to estimate the probability of data loss following a node failure.

The Relation of RPO to Replication Latency

In systems using replication to keep databases in synchronism, there are two forms of data replication that might be used – synchronous replication and asynchronous replication. If

synchronous replication is used,[1] there is no data loss following a node failure since no change is made to any database copy in the application network unless that change can be made to all database copies.

However, if asynchronous replication is used,[2] changes are replicated after the fact from a change queue. Therefore, should a node fail, changes that are still waiting to be replicated may be lost. This leads to lost transactions that may have a significant impact on corporate finances or on operations. Therefore, it is common that an impact analysis be done on each critical application and that a limit be placed on the amount of data that may be lost following a node failure. This limit is the RPO, and it typically varies from application to application.

In calculating RPO, we are dealing with a queuing system. Changes are queued and are processed on a first-in, first-out basis by a server, the replication engine. The replication engine extracts the next change from the change queue, sends it to the target system, and applies it to the target database. The interval from the time that the change is entered into the change queue to the time that it is applied to the target database is the response time of the server. We call this response time the *replication latency* of the replication engine.

If we assume that any change in the change queue or in the replication engine server at the time of a node failure is lost, then knowing the distribution of the replication latency time will let us make a statement about the probability of data loss.

Estimating RPO

The measurement of data loss following a node failure in actual practice can be quite complex. Unless this measurement can be made, there is no assurance that the RPO specification can be met.

This estimate can be achieved via two different methods:

- The replication channel can be modeled.
- The replication channel can be measured.

Modeling the replication channel can be very complex if even possible, and the model must be verified by measurement. Therefore, the replication channel must be measured anyway; so why not simply use those measurements to calculate the RPO and avoid possible modeling errors? This is the technique to be described below.

The procedure depends upon the use of a very powerful probability distribution – the Gamma distribution, which we first introduce.

The Gamma Distribution

The Gamma distribution comes with a compelling history. In the early days of telephony, the Erlang distribution, developed by A. K. Erlang, was used to determine the number of telephone calls that might be waiting for telephone operators. This work has been expanded to determine the distribution of waiting times in queuing systems in general, such as those of a replication channel. The resulting probability distribution is the Gamma function. It has been shown to give surprisingly close results to actual response times measured in practice or by simulation.

[1] Synchronous Replication, *Availability Digest*; December 2006.
[2] Asynchronous Replication Engines, *Availability Digest*; November 2006.

The Gamma distribution[3] provides the probability that a variable will have a value less than a specified value – just what we need. It depends upon only one parameter represented by R, which is the ratio of the square of the distribution's mean to its variance.

Since we are interested in the distribution of replication time, for our purposes *R* is given by

$$R = \frac{\left(\overline{T_r}\right)^2}{\text{var}(T_r)} \qquad (1)$$

where

T_r is the replication time

$\overline{T_r}$ is the mean (average) replication time

var(T_r) is the variance of the replication time, T_r .

Thus, if we can determine the mean and the variance of the replication time (that is, replication latency), we can make a statement about the probability that the response time will be less than a certain value. It can be shown that both the mean replication time and its variance are a function of the load on the replication channel. This will become important when we make our measurements.

The mathematical representation of the Gamma distribution is not very palatable (see page 439 of the above footnote referencing Martin's book). Fortunately, Excel comes to the rescue with its GAMMADIST function. For our purposes, we use

$$\text{Probability } (T_r < T_m) = p = (\text{GAMMADIST}(zR, R, 1, \text{TRUE}) \qquad (2)$$

where *z* is the ratio of the maximum specified replication time to the mean replication time:

$$z = \frac{\text{maximum replication time}}{\text{mean replication time}} = \frac{T_m}{\overline{T_r}} \qquad (3)$$

and T_m is the maximum specified response time.[4]

This distribution is shown in Figures 1a and 1b. Figure 1b explodes the higher probabilities.

Measuring Replication Latency

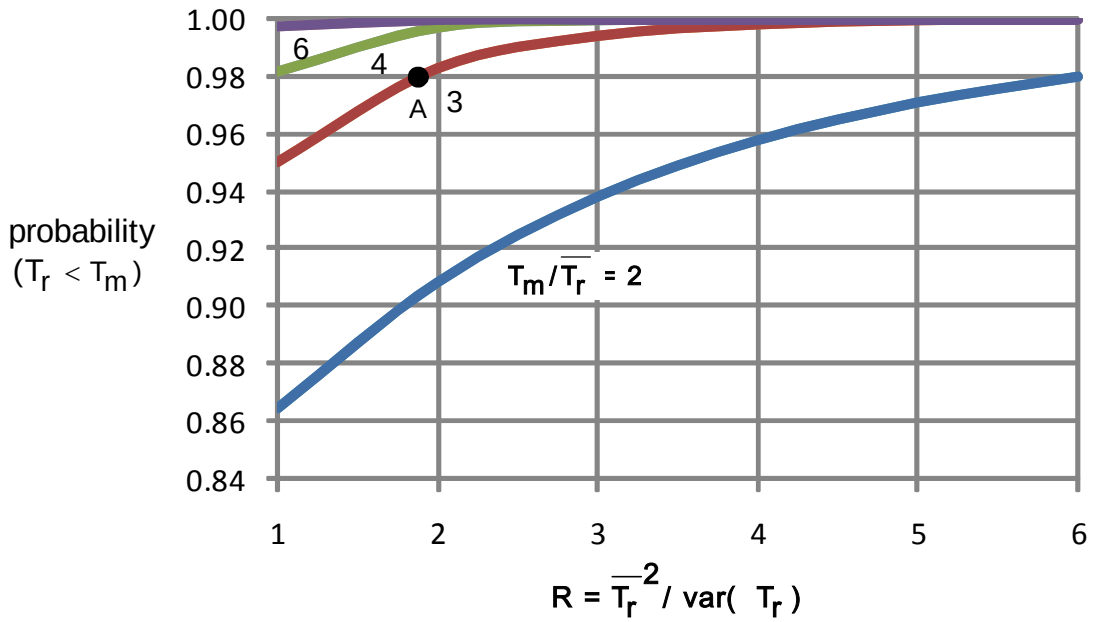
In order to measure the actual replication latency, we must somehow instrument the replication engine. To do this, we tag each change with the time that it was generated as well as the time that it was applied to the target database. The difference between these times is the replication latency for that particular change.

Let us assume that our RPO specification is given by

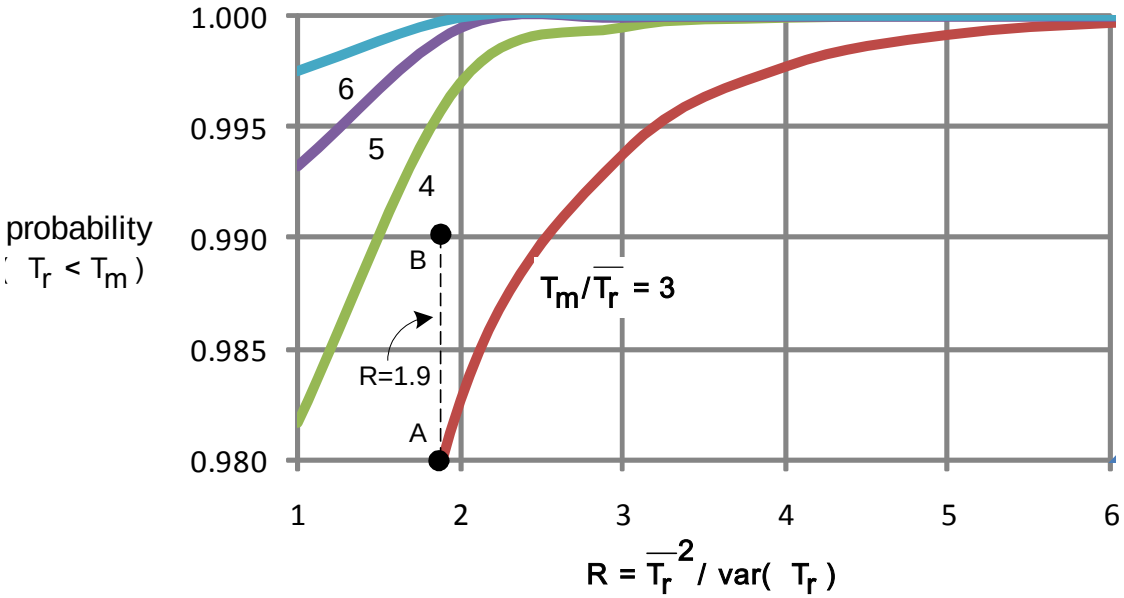
> "99% of node failures will result in no more than 300 milliseconds of lost data at 100 transactions per second."

[3] James Martin, Chapter 31, Queuing Calculations, pp. 438-444, *Systems Analysis for Data Transmission*, Prentice-Hall; 1972.

[4] The third parameter, "1," returns the standard Gamma distribution. The parameter "TRUE" returns the cumulative distribution. "FALSE" returns the probability density function. Excel also provides an inverse Gamma function, which provides *zR* = GAMMAINV(*P*, *R*, 1).



Allowable Variance to Meet RPO
Figure 1a



Allowable Variance to Meet RPO
Figure 1b

We generate a transaction load of 100 transactions per second and measure the replication latency of each change that is replicated. Typically, this would be thousands of changes that would be analyzed programmatically. However, to illustrate the calculation, let us take just five samples of replication latency, which are:

	Measured Replication Latency (msec.)
	38
	144
	53
	41
	224
Total	500
Mean	100 msec.

Our average replication latency is 100 msec. The next step is to calculate the variance of these measurements. The variance is the average of the squares of the deviations of each measurement from the mean:

Measurement (msec.)	**Deviation from Mean (msec.)**	**Deviation^2 (msec.^{2})**
38	-62	3,844
144	44	1,936
53	-47	2,209
41	-59	3,481
224	124	15,376
	Total	26,846
	Variance	5,369

The values of our parameters are thus:

T_m = the maximum specified response time (from the RPO specification) = 300 msec.
$\overline{T_r}$ = the mean (average) replication time = 100 msec.
var(T_r) = the variance of T_r = 5,369.

Then, from Equations (1) and (3), we find that:

$$R = \frac{\left(\overline{T_r}\right)^2}{\text{var}(T_r)} = \frac{(100)^2}{5{,}369} = 1.86 \approx 1.9$$

$$z = \frac{T_m}{\overline{T_r}} = \frac{300}{100} = 3$$

This is plotted as point A on Figures 1a and 1b. We see that we do not meet the RPO specification. In fact, only 98% of failures will result in less than 300 msec. of lost data, not 99% as required by the RPO specification.

How can we fix this? We obviously have to speed up the replication engine. But by how much?

Excel comes to the rescue with the inverse GAMMA function referenced in footnote 4:

$$zR = \text{GAMMAINV}(p, R, 1) \qquad (4)$$

This function makes a very simple statement. If we know the probability *p* and the parameter *R*, we know the quantity *zR*. We know the probability *p* that we are trying to achieve – 99%. What about *R*? Interestingly, *R* remains the same (at least to a first approximation) if we simply speed up the replication engine. The distribution of response times stays the same – they just get faster. The mean and deviations both decrease proportionately, leaving *R* the same. Therefore, we can use the inverse Gamma function, Equation (4), to calculate the value of *z* that we must achieve. Using Excel, we find:

$$zR = \text{GAMMAINV}(0.99,\ 1.86,\ 1) = 6.45$$

$$z = \frac{6.45}{R} = \frac{6.45}{1.86} = 3.46 = \frac{T_m}{\overline{T_r}}$$

$$\overline{T_r} = \frac{T_m}{z} = \frac{300}{3.46} = 87 \text{ msec.}$$

This is plotted as point B on Figure 1b.

This leaves us with two tasks. The first task is to determine what components in the replication channel can be upgraded to pick up 13 msec. (perhaps a faster communication link will do the trick).

The final task is then to confront management with the choice of the cost of upgrading the replication channel or the cost of a probability of 2% of unacceptable lost data rather than a 1% chance.

Summary

By using the Gamma distribution curves in Figures 1a and 1b, you can determine whether you can meet an RPO specification based on straightforward measurements of replication latency. If your current system does not meet the specification, you can determine the specification that it can meet.

Alternatively, you can determine how much you have to speed up the replication channel in order to meet the specification. A cost/benefit decision can then be made as to whether to spend the money to speed up replication or to accept a reduced RPO specification.