

## **Software Testing Technical Note 2004-01**

**Load Testing - Issues that can arise when  
think time is reduced or eliminated.**

The logo for SQC Technology Ltd. is a dark blue triangle pointing to the right. Inside the triangle, the letters 'sqc' are written in a lowercase, orange, sans-serif font.

**sqc**

[www.sqc.co.uk](http://www.sqc.co.uk)

SQC Technology Ltd.  
Coton Park House, Linton, Swadlincote,  
DE12 6RA, UK.

## Table of Contents

Table of Contents .....	2
<b>1. Introduction .....</b>	<b>3</b>
1.1 Load-Testing .....	3
1.2 Getting More Load from the Load Generator .....	3
1.3 The Fidelity of the Simulation .....	4
<b>2. An Example Scenario .....</b>	<b>5</b>
2.1 The Example Transactions .....	5
2.2 Transaction Execution Time .....	6
2.3 Load Generator Channel Requirements .....	6
<b>3. Eliminating Idle Time Whilst the Channel is Disconnected .....</b>	<b>7</b>
3.1 The Proposal - Eliminating Operator Response Time .....	7
3.2 The Consequences .....	7
3.3 Is a Reduction in the Number of Concurrent Transactions a Problem? .....	8
3.4 Alternatives - Additional Concurrent Transactions Without Idle Channels .....	10
3.5 The Bigger Picture .....	12
<b>4. Eliminating Idle Time Whilst the Channel is Connected .....</b>	<b>13</b>
4.1 The Proposal - Eliminating Client Processing Time .....	13
4.2 The Consequences .....	13
4.3 Are the Consequential Changes a Problem? .....	14
4.4 'Safe' Approaches to Eliminating Channel Idle Time in the Connected Periods .....	16
<b>5. Conclusion .....</b>	<b>18</b>
Author Biography .....	19
SQC Technology .....	19

# 1. Introduction

## 1.1 Load-Testing

Load-Testing is testing that applies a simulated workload to the system. This workload simulates a scenario that is based on the expected real-world usage of the system. Typically it simulates multiple users submitting transactions at a high rate. There are various reasons for performing this type of testing, the two main ones being assessment of the response-time / throughput (Performance-Testing) and assessing the integrity of the system under load (Stress-Testing).

## 1.2 Getting More Load from the Load Generator

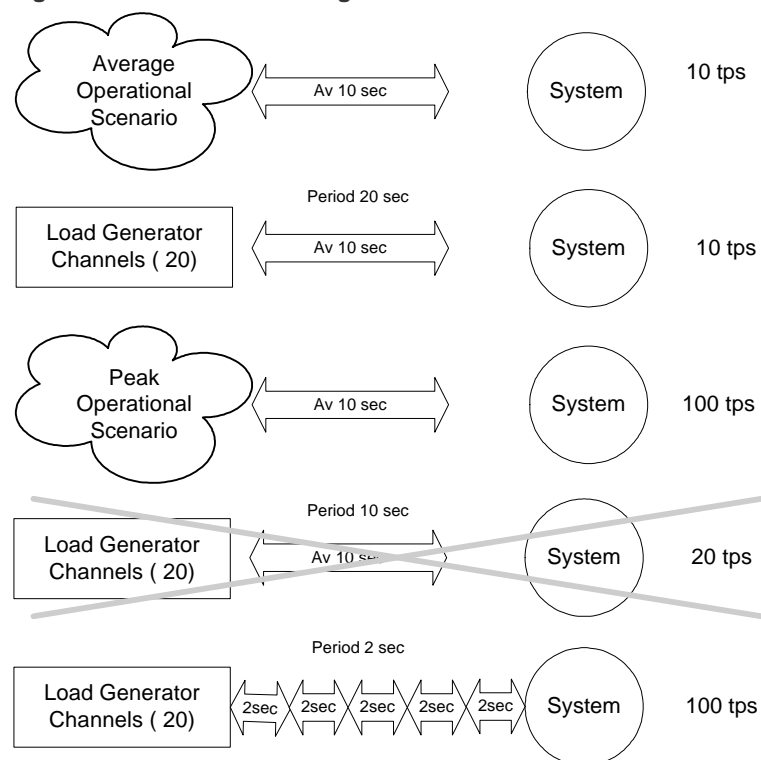
Generally load generators have a limit on the number of simultaneous client / users they can simulate. There may be a desire to get more load from the generator than a simple simulation of a client / user would allow. For example peak traffic rates on a web-site can be ten times the average traffic rates. Simulating the peak traffic scenario requires ten times more output from the load generator than simulating the average traffic scenario.

A transaction, a sequence of interactions with the system, occupies one channel of the load generator for the duration of the transaction. The highest average transaction rate a load generator channel can sustain is governed by the average length of time it takes to deliver a transaction. The relationships are:

$$\begin{aligned} \text{Average-Channel-Tran-Rate(t/sec)} &= 1 / \text{Average-Tran-Duration(sec)} \\ \text{Average-Lgen-Tran-Rate(t/sec)} &= \text{Average-Channel-Tran-Rate(t/sec)} \times \text{Num-Channels} \end{aligned}$$

A technique often used to obtain more load is to accelerate the client / user responses. Rather than waiting for the expected real-time delay, 'the user thinks for 8 seconds', a much shorter time is used. This reduces the overall transaction time and increases the number of transactions per second that a generator channel can produce. See figure 1 on page 3.

Figure 1 Simulation of Average and Peak Scenarios



The illustration shows that a 20 channel load generator can simulate the average operational scenario ( 10 t/sec ). Each channel performs a transaction once every 20 seconds.

When simulation of the peak operational scenario is attempted the best the generator can achieve without shortening the transaction time is 20 t/sec. However if client / user response times are accelerated reducing the transaction duration to 2 seconds then the load generator can achieve the target 100 t/sec.

The accelerated simulation is delivering the level of external demand defined as the peak operational scenario but the average transaction duration is shorter.

### 1.3 The Fidelity of the Simulation

For the results of Load-Testing to be valid the following two conditions must be true:

- ! The scenario must be an accurate representation of the real-world situation being addressed. For example it may be a representation of 'Expected worst case mean transaction arrival rate measured over a 1 hour period.'
- ! The fidelity of the scenario simulation implemented in the load generator must be good. That is all of the significant characteristics of the scenario must be matched by the characteristics of the simulation.

This note is concerned with the fidelity of the simulation of a scenario when acceleration of client / user response is being used to increase the load that can be delivered.

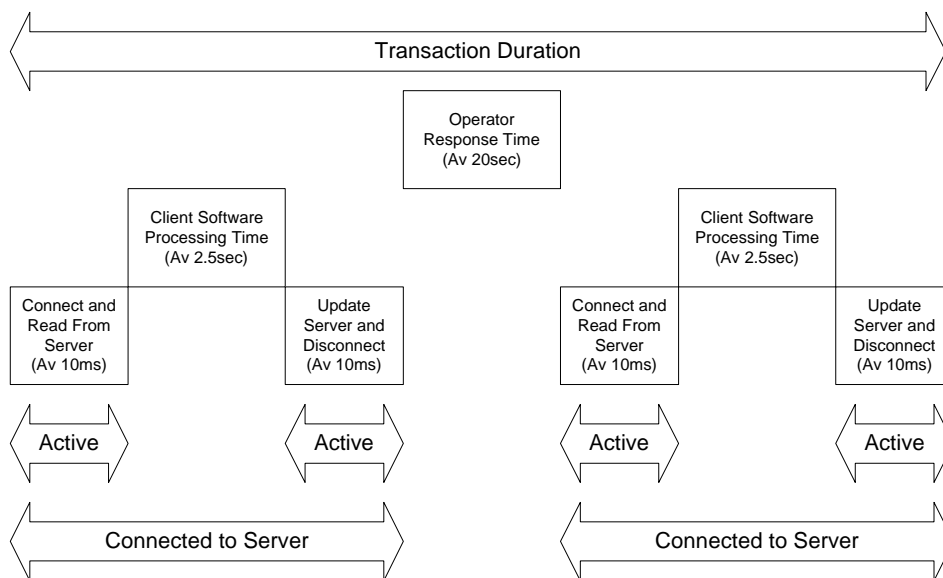
The example shown in figure 1 on page 3 illustrates the use of acceleration to increase the available load. A simulation of a scenario that uses acceleration can have a different impact on the system to one that uses more channels without accelerating the client / user responses. Acceleration can create discrepancies between the characteristics of the scenario and the characteristics of the simulation. The types of discrepancies that can arise and, where appropriate, approaches to dealing with these discrepancies are described in this note.

## 2. An Example Scenario

### 2.1 The Example Transactions

A simple example will be used to illustrate the side effects arising from reducing or eliminating the client / user delays in order to reduce the duration a channel is occupied by a single transaction. The transaction sequence is shown in figure 2 below.

Figure 2 Example Transaction



The transaction involves the client connecting to the server for two separate periods. These are separated by a period during which the client is not connected to the server. During this middle, unconnected, period the client waits for an operator action. The average duration of this period is 20 seconds.

Each of the connected periods involves (a) the client connecting to the server and obtaining data from it, (b) client processing the data obtained from the server and (c) the client updating the server and closing the connection. Client processing takes 2.5 seconds. Each of the interactions between the client and the server takes 10ms and occupies the server processor for 10ms.

The peak arrival rate of these transactions is 72000 t/hr or 20 t/sec.

## 2.2 Transaction Execution Time

The execution time for a single transaction is 25.04sec, the sum of the individual stages given above. However under load conditions contention occurs during the server interactions. This will extend the execution time.

With the example transactions arriving at a rate of 20t/sec the resulting arrival rate for interactions with the server is 80i/sec. Assume that the amount of server processing time required per interaction is independent of the number of interactions waiting to be processed. Now a queue model shows that the response time (not the processing time) for the interactions will be 50msec with an average of 4 in progress at the same time.

This gives the following times for the interaction:

Component	Average Time
Phase-1 Connect and Read	50msec
Phase-1 Client Processing	2.5sec
Phase-1 Update and Disconnect	50msec
Operator Response Time	20sec
Phase-2 Connect and Read	50msec
Phase-2 Client Processing	2.5sec
Phase-2 Update and Disconnect	50msec
<b>Total:</b>	25.2sec

## 2.3 Load Generator Channel Requirements

The peak transaction arrival rate is 20t/sec. Assuming an as is simulation of the transaction, with the load generator channel pausing to simulate the client processing and the operator response time, then the channel will be occupied for 25.2 seconds. The number of load generator channels required is 20 x 25.2 giving 504 channels.

Some additional allowance will need to be for generator overhead and to compensate for slow responses from the system. If the system response is slow then the time the channel is occupied will increase and the rate at which it submits transactions will fall. Thus as the system slows the arrival rate may fall allowing the system to recover.

This self regulation would not happen in an open system with a large number of clients. The arrival rate would remain the same possibly causing a further deterioration in response or even a catastrophic system failure. Extra capacity is required to prevent the load generator letting the system off the hook<sup>1</sup>.

So, given the above, a provision of 550 or 600 channels could be required to simulate the peak transaction arrival rate.

1 However when the extra capacity is present it also requires active regulation of the arrival rate when the system response is not slow in order to avoid exceeding the target rate.

### 3. Eliminating Idle Time Whilst the Channel is Disconnected

This section looks at eliminating the periods when the load generator channel is idle, in a time delay, and is not connected to the system under test. It explores the impact that eliminating these periods can have on the fidelity of the simulation of the test scenario. Approaches are outlined that may enable the idle time to be reduced whilst maintaining the fidelity of the simulation.

#### 3.1 The Proposal - Eliminating Operator Response Time

In the basic simulation, described in the previous section, the load generator channel sits idle, disconnected from the server, for an average of 20 seconds during each transaction. On the face of it there is no activity in the server related to the transaction. Let us reduce this delay. We will not take it out completely, the server may need to settle, but we will reduce it to 1 second, still an eon for a modern server.

Now with the operator response time in the simulation reduced to 1 second what benefits arise? Well the average period taken for the load generator to apply a test transaction now falls to 6.2 seconds. The generator channel is occupied for 6.2 seconds. Given this the nominal number of channels required to achieve a 20t/sec arrival rate is now  $20 \times 6.2$  giving 124 channels. Allowing for overheads etc. the load generator system can now apply the required load with around 150 channels rather than the 600 originally required.

#### 3.2 The Consequences

What are the consequences of reducing the operator response time? Are there any unforeseen side effects? The possibilities are discussed below.

##### Server Requests - No Change

Could there be a change in the level of demand on the server, that is a change in the intensity of the requests arriving from clients? Well there are still 20t/sec arriving and each still results in 4 server interactions during its life. The result is an average of 80 requests per second arriving at the server. Some from transaction just starting, some from older transactions.

This request rate is the same as it was in the original simulation with an operator response time of 20 seconds. There is no significant change here.

##### Connection Events and Number of Concurrent Connections - No Change

Connection creation events still occur at an average rate of 40 per second. Connection deletion events still occur at an average rate of 40 per second. The average duration of a connection remains the same at 2.6 seconds.

What about the number of concurrent connections to the server? Well firstly how is this estimated? The estimate is based on the following rationale. During the life of any connection the average rate at which additional new connections arrive at the server is the same as the average rate at which existing connections leave the server (assuming the server is not saturated). Hence the average number of connections in the server at the start of a given connection is equal average number at the end. For a connection the number of connections that arrive during its life is equal to the time the connection exists multiplied by the connection arrival rate. This gives the average number of connections in the server.

Now neither the average lifetime of the connection (2.6 seconds) or the connection arrival rate (40 per second) have changed. Hence the average number of connections in the server remains unchanged (104 connections).

**Number of Concurrent Transactions - Changed!**

The rationale given in the previous section for the number of concurrent connections can be applied to estimating the number of concurrent transactions. We calculate the number of concurrent transactions by multiplying the duration of a transaction's lifetime by the arrival rate. The figures for the two cases are given below:

	Original Simulation	Op Resp Time Eliminate
Average Transaction Duration	25.2sec	6.2sec
Transaction Arrival Rate	20t/sec	20t/sec
Average Number of Concurrent Trans	504	124

The average number of concurrent transactions has fallen from 504 to 124. Note that these figures equate to the nominal number of load generator channels. Now this is quite an intuitive relationship as the nominal number is calculated as being the number required to achieve the target rate when each does a transaction and then immediately starts the next.

So the number of concurrent live transactions has fallen. Is this an issue? This is discussed in the next section.

**3.3 Is a Reduction in the Number of Concurrent Transactions a Problem?**

As described above, reducing the operator response time in the simulation has reduced the average number of concurrent transactions. So is this a problem? Firstly the answer depends upon the nature of the system under test and on the way it is implemented. Secondly if the level of concurrency does impact on the system's behaviour then whether this is an issue for a given test will depend upon the objectives of the test.

**Does it impact on the system?**

Factors that could make this an issue include:

- ! Does the system create any data structures that exist for the duration of the transaction and are then disposed of at the end of the transaction?
- ! Are there any functions within the system that select and operate on the set of live transactions whilst ignoring others?
- ! Do live transaction lock any resources and hold them through the operator response period<sup>1</sup>?

The amount of resources in use for temporary data structures can influence the behaviour of the system. There may not be enough resource available to support the structures for 500+ concurrent transactions but this problem may not be seen with 124. Less directly the existence of structures for a large number of transactions may increase the processing time of some functions. An example of this could be selecting a set of records from the database. This may cause performance degradation. This effect may not be detectable with the smaller number of transactions, especially if the degradation is not linear.

Functions that operate on the set of live transactions have more work to do when there are more transactions in the system. This could affect the time they take to operate, possibly even the amount of time per transaction they must process. It could also affect the amount of resources, say memory, they require during their operation. Again with fewer transaction issues could be missed. A periodic audit function could soak up processor time. A clean-up routine could run out of memory and fail.

---

1 Resources that are locked whilst the client is connected but then released before or at disconnection are not an issue. The average number of connected clients, and hence transactions in a connected state, has not been changed.

If live transactions lock resources during the operator response period then there is a threat of resource exhaustion or possibly an increased likelihood of a collision over a resource. Two types of resources can be considered. Anonymous pools of resources used to support the processing of a transaction and application domain resources.

For anonymous resources the danger is exhaustion. If there are too few to support the number of live transactions present then a rejection will occur. If this is dealt with it will trigger different planned behaviour which may have bugs or may affect processing times. If it is not dealt with the consequences could be catastrophic.

For application domain resources, say the stock level of an item, the risk is another transaction requiring the resource whilst it is locked. Obviously the probability of conflict increases as the number of resources locked increases. The number of resources locked will be proportional to the number of transactions in the system. Thus the probability of having a conflict and triggering the response increases as the number of transactions increases. Also a transaction that has a conflict may use more processor time than one that does not. The frequency of collisions can affect the processor utilisation.

### Is it an issue for this test?

The previous section has outlined potential consequences of the average number of concurrent transactions being lower when the operator response time is reduced to 1 second. If there is a possibility that some of these issues do exist then must all tests achieve the higher average number of transactions if they are to be valid?

The end results of the problems described above fall into two broad categories - problems that affect the amount of processing required and problems that cause malfunctions. Taking the two uses of load testing, Performance-Testing and Stress-Testing, it can be argued that:

- ! If there is a potential of affecting the processing requirements then Performance-Testing needs to operate with the higher number of live transactions.
- ! If there is a potential for causing malfunctions then Stress-Testing needs to operate with the higher number of live transactions.

Now obviously there is always some potential for problems. The issue is the level of risk. This can only be established from reviewing the system architecture and from experience with the system. If the operation of system is clear then an assessment may establish that there are no tests that require the higher number of concurrent live transactions. In this case it may be acceptable to ignore the issue.

On the other hand if the operation of the system is unclear or an assessment identifies one or more tests that require the higher number of transactions then we need the ability to achieve this. Does this mean going back to the higher number of load generator channels? Not necessarily. The next section outlines approaches that allow the higher number of concurrent live transactions to be achieved whilst leaving the operator response time at 1 second.

We shall see that there are methods that provide the higher number of concurrent live transactions without requiring more load generator channels. Given this, it makes sense to use these approaches in all testing under load. Mistakes made when assessing whether there is a potential problem could cause testing to miss faults. Always achieving the realistic number of concurrent live transactions will prevent this<sup>1</sup>.

---

1 However there are two types of test where it is probably appropriate to minimise the delays at the expense of the fidelity of the simulation. One is where the endurance to a cumulative number of transactions is being assessed. The idea being to do a very large number of transactions. Delays can extend the time taken to apply the required number of transactions. The other case is where high instantaneous load is to be applied to the system. This is the 'simultaneous' arrival of a large number of transactions.

### 3.4 Alternatives - Additional Concurrent Transactions Without Idle Channels

#### Create a Number of Static 'Live' Transactions

Perhaps the simplest approach to the problem is to create a number of transactions in the system at the start of the test. These transactions never enter the third stage. They remain 'live' forever. So in this example, prior to starting the main test run, 380 transactions are created. Each is left in the awaiting operator response state. Now the main test run adds, on average, 124 live transactions giving a combined average of 504.

Problem solved! Well possibly. The additional transactions are static. The volume of temporary data structures is increased but a large proportion is static. Changes are restricted to the smaller section of the structures. This may mask problems where structures fragment. It may mean updates are always done on cached areas and so reduce processing demands. On the other hand resources held by these 'live' transactions are held permanently. This may cause problems where a clash occurs. Another transaction may be held forever disrupting operation.

Adding additional 'live' transactions is an improvement over the original situation. However the static nature of the transactions is not ideal. Alternative approaches, that avoid permanent transactions, are discussed below.

#### A Changing Set of Additional 'Live' Transactions

The next approach dedicates a few additional load generator channels to operating on the additional 'live' transactions. As in the previous approach an initial set of 380 transactions is created. Then, rather than leave the same transactions there forever, the extra channels work through them completing one and starting another. A channel takes 6.2 seconds to do a create transaction operation followed by a complete transaction operation. So with 10 channels it will take around 4 minutes to change all of the additional transactions. The average life of an additional transaction will be 4 minutes, compared with 6.2 seconds for the other test transactions and the 25.2 seconds predicted for the scenario being simulated.

This is an improvement but two points should be noted. Firstly though the set is changing the rate of change is nowhere near as high as the rate predicted for the scenario. Secondly the additional channels are submitting some extra requests to the server. This may not be significant but if it is then the two submission rates need to be co-ordinated to ensure that together the correct rate is achieved. This complicates the testing process.

#### Maintaining the Correct Transaction Lifetime Without Idle Channels

The original simulation scheme had four desirable characteristics:

- ! All transactions are applied in the same way.
- ! The average lifetime of a transaction matches that predicted for the scenario being simulated ( 25.2sec ).
- ! The average number of concurrent live transactions matches the number predicted for the scenario being simulated (504).
- ! The workload imposed on the server matches that predicted for the scenario being simulated ( 80i/sec ).

The downside with the original simulation scheme was idle channel time and hence a requirement for 504 channels. Ideally we want to achieve these characteristics without idle time and with 124 channels. Ways to achieve this are described below.

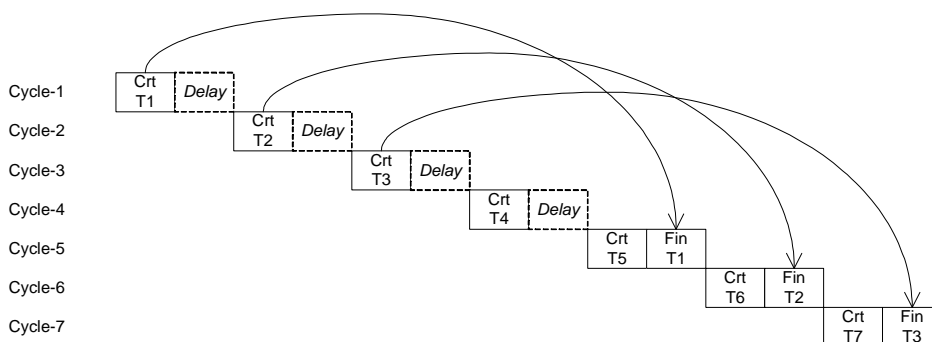
This goal can be achieved if the load generator channels are treated as worker channels that can be allocated to perform individual tasks for a transaction. The channel is no longer dedicated to one transaction throughout the lifetime of that transaction. Two schemes that avoid dedicating a channel to a single transaction are outlined in the next two sections. The first, and simplest one, depends on all tasks for a given transaction being performed by the same channel. This has certain limitations and so a second more flexible scheme is also described.

**Interleaved Operation Within A Single Channel**

In this scheme a generator channel interleaves operations on different transactions. On each cycle it has a slot to start a transaction and a slot to end a transaction. The main difference, when compared with the original scheme, is that the transaction it stops is not the one it has just started. Instead it stops one started earlier, in this example one started 4 cycles earlier.

This is illustrated in figure 3 on page 11. The illustration includes the start-up period where for the first 4 cycles there are no transactions to stop. When there is no task to be done the channel delays for 2.6 seconds. A similar null operation is required for the start action when the channel is shutting down.

**Figure 3 Interleaved Operation Within A Single Generator Channel**



The characteristics of this scheme are:

- ! All transactions are applied in the same way.
- ! The average time of a transaction will be 26 seconds ( 5.2 sec x 5 ).
- ! At the specified arrival rate ( 20 t/sec ) the average number of concurrent live transactions will be 20t/sec x 26sec, giving 520 transactions.
- ! Nominally the simulation requires 112 load generator channels to supply the required 20t/sec.
- ! With 112 channels in operation the workload imposed on the server is that predicted for the scenario, 80i/sec.

This scheme is fairly simple and offers a better simulation of the scenario whilst using slightly fewer load generator channels. However there are a small number of issues with this scheme. In most cases these issues will be not be significant but it is worth identifying them.

Firstly it should be noted that the average transaction lifetime is 26 seconds not 25.2 seconds. This is because it is an integer multiple of the average time taken to perform the two operations. With the example chosen it turns out that a close enough average can be achieved without any wasted time in the channel. On the other hand if the operating time had been 8 seconds then it may have been necessary to introduce a 2.2 second pause every third cycle. If this were the case then the number of generator channels needed to achieve the target transaction rate would rise when the pause was added.

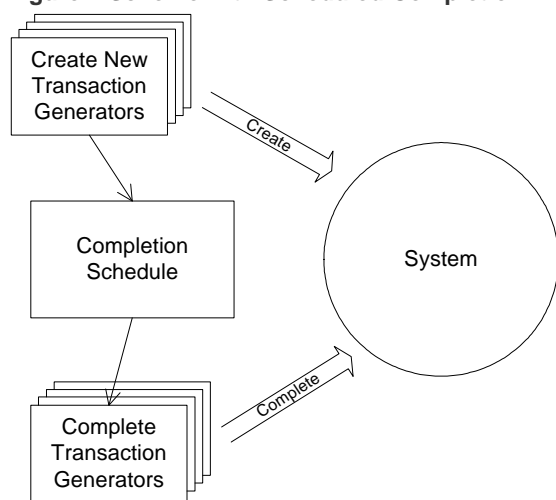
The second point is that the spread of transaction lifetimes is totally dependent on the spread of times taken to perform the create and complete interactions. We can not directly simulate the expected spread of operator response times. For example we may expect some operator response times to be as low as 1 second. This can not occur with this scheme for simulating the scenario. The lower limit will be four times the minimum time taken to perform a cycle. An alternative scheme is needed if the distribution of operator response time is to be independent of the time taken to perform the creation and completion operations.

### Using Scheduled Completion Times

In this scheme the lifetime of a transaction is set as the transaction is created. Say, for example, that the scenario being simulated states that operator response times are uniformly distributed in the range 1 second to 39 seconds, giving a mean of 20 seconds. Then as a transaction is started a random number generator is used to produce a response time in this range. This is then added to the time the creation operation ends to give the time that the transaction is scheduled for completion.

The completion times of each transaction are held in a shared data pool. Generator channels that are due to perform a completion operation access this pool to identify transactions that are now due for completion. The channel takes the one with the earliest completion date and performs this operation. The simplest way to implement this scheme is to use one set of channels for creation and another set for completion. This is illustrated in figure 4 on page 12<sup>1</sup>.

**Figure 4 Scheme with Scheduled Completion**



There are a small number of technical issues to consider if this approach is to be adopted. They relate to both the nature of the system / transactions and to implementing this approach in the load generator.

Using a different channel to complete the transaction may encounter problems if the system can identify the client and records information about the client that created the transaction. The system may not accept another channel completing the transaction unless the channel can spoof the system making it think that the same client is completing the transaction. One attribute of the client identity that could cause problems is the IP address.

Implementing this approach within the load generator requires that channels can share the completion schedule data structure in a secure and efficient way. There are many writers and many

readers of this data. This is a classic computer systems implementation problem. It can be addressed by the use of appropriate implementation techniques for dealing with concurrency in data structures.

### 3.5 The Bigger Picture

This example of eliminating operator response time has highlighted the issues around the number of concurrent live transactions in the system. The number present is independent of the number of connected clients. It is a quantity not easily recognised as a key attribute of the system state.

Within a system there could be many other entities from both the application domain and implementation domain where the size of the population could affect system behaviour. It could be the number of instances of an entity or the number of instances in a given state. For example entities could include the number of customer accounts, the number of orders in progress or the number of transactions in progress. An example of an entity in a given state is the number of orders in the completed state.

To allow effective testing an attempt should be made to identify the significant attributes of the system state. This will involve a study of the application domain and the operation of the system. Having identified the significant attributes predictions of their values can be made for each operating scenario. The proposed simulation(s) of the scenario can then be checked to see how well the system attributes will match those predicted for the scenario<sup>2</sup>.

1 Note that with this scheme the completion channel work rate is controlled by the create channel work rate. With the creation generators operating at 20t/sec the completion generators will inevitably operate at this rate.  
 2 There is a grey area here where the application of work load and the application of data volume overlap. There is not an absolutely clean distinction between Load-Testing and Volume-Testing.

## 4. Eliminating Idle Time Whilst the Channel is Connected

In section 3 elimination of delays whilst the load generator channel is disconnected from the system was discussed. In this example that equated to eliminating the operator response period. The potential impact of this was identified and ways were outlined that avoid idle time in load generator channels whilst maintaining the fidelity of the load simulation.

Now suppose that the load generator still does not have enough channels to generate the peak load. Can further optimisation be done and if it is done what is the impact on the fidelity of the simulation? What further channel idle times can be eliminated?

As section 3 dealt with the load generator channel being idle whilst not connected to the system any further reductions must look at idle time whilst the channel is connected to the system. In the example used in the paper this means looking at the client processing time that occurs in each period that a channel is connected to the system. This will be discussed in this section.

### 4.1 The Proposal - Eliminating Client Processing Time

The possibility of further optimising load generator channels usage is to be investigated. Perhaps because we are dealing with a very high arrival rate and the number of channels is still huge. Perhaps because 120 to 150 channels is still more channels than are required to simulate the average load.

The proposed method of doing this is to reduce the simulated client processing time from 2.5 seconds to 0.5 seconds. This will eliminate 4 seconds of idle time from the generator channel cycle. Now assuming that either the interleaving or schedule completion scheme is being applied so that there are no operator response time delays in the load generator then this change reduces the generator cycle time from 5.2 seconds to 1.2 seconds.

The benefits from this? The nominal number of channels required to apply the required transaction arrival rate now falls to  $1.2\text{sec} \times 20\text{t/sec}$  giving 24 channels. Thus with both operator response time and client processing time eliminated the required transaction arrival rate can be achieved with 24 channels as opposed to 504 when the original simulation method was used.

### 4.2 The Consequences

What are the consequences of reducing the client processing time? Are there any unforeseen side effects? The possibilities are discussed below.

#### Server Requests - No Change

Is there any change in the intensity of requests arriving from the clients at the server? Assuming there are 24 channels in use then there will be 20t/sec commencing. Each transaction will perform 4 interactions with the system during its lifetime. So there are still an average of 80i/sec with the system. The intensity of demands on the system has not changed.

#### Connection Events, Connection Duration and Number of Concurrent Connections - Changed!

There are two connection events per transaction. Transactions are still arriving at 20t/sec. So there are, on average, 40 connection events per second. This rate is the same as the rate when the simulation contained a client processing time of 2.5sec. There is no significant change in the occurrence of connection events.

The average connection duration is equal to the interaction durations plus simulated client processing time. This has now changed from 2.6 sec to 0.6sec.

What about the number of concurrent connections. As with live transactions the average number of concurrent connections is equal to the average lifetime of a connection multiplied by the average arrival rate. As described above the average arrival rate is 40c/sec. The average life of a connection is 0.6sec. So the average number of concurrent connections is 24. If the same calculation is done for the simulation with the original client processing time then the prediction is that the number of concurrent connections will be 104. There has been a substantial reduction.

#### Number of Concurrent Transactions - It Depends!

If the simple, one channel dedicated to a transaction for its duration, approach is in use then the effect is the same as reducing the operator response time. The average transaction lifetime is reduced with the consequences discussed in section 3.2 on page 7.

If the interleaved scheme is in use then the duration of each operation changes which will change the average transaction lifetime. This could be addressed by adjusting the interleaving scheme.

If the scheduled completion time scheme is in use then changing the time taken to start or complete a transaction can be compensated for by adding the average reduction in the times taken for the operations to create and complete the transaction onto the selected operator response time. When this is done the scheduling ensures that the average duration of a transaction is unchanged.

#### Number of Transactions in Creating or Completing State - Changed!

The average number of transactions in the creating state is the product of the average time taken to perform the creation operation and the transaction arrival rate. The average time taken to perform the creation operation has fallen, we have reduced the client processing time in the simulation, and so the average number of transactions in the creating state has fallen from 52 to 12.

By a similar argument it can be seen that the average number of transactions in the completing state will fall. Again, in our example, this will be from 52 to 12.

On the other hand if the average transaction lifetime has been maintained, using one of the techniques discussed in section 3.4 on page 10, then the average number of transactions in the middle disconnected state will have risen from 400 to 480. Note that this rise balances out the reductions in the other two states as a transaction must be in one of the three states.

### 4.3 Are the Consequential Changes a Problem?

#### Does the reduction in the average connection duration impact on the system?

In the example we are discussing the original average connection time was 2.6secs. It has now reduced to 0.6sec. This is unlikely to have a direct impact on the operation of the system. In this example any impact is likely to be indirect, via the reduction in the average number of concurrent connections, something that is discussed in the next section.

However in other situations a reduction in the average connection time could have a more direct impact. If the operations across a client-server connection extend over a longer period then timeouts may come into effect causing additional work. The client-server interaction is supported by layers of software. For example it may be carried in HTTP, which in turn is carried on TCP. In each of these supporting layers there will be connections. If the top level client connection is dormant for a time then timeouts may occur in the lower layers. The supporting connections may be destroyed. When activity between the client and server resumes the supporting connections must be re-established.

Both connection timeouts and re-establishing connections can incur a substantial processing overhead. On timeouts data structures have to be destroyed, on re-connection rebuilt. This imposes a delay on the specific connection involved and also reduces processor time available for other work. If connections are long and regularly timeout then a lot of processor time can be soaked up. The scenario being simulated may be one in which timeouts would occur. If the simulation reduces the average connection time then these timeouts may not be triggered when the simulation operates. In this case the system performance seen may not reflect the performance that would occur when the system encounters the scenario in live operation.

### Does the reduction in the average number of concurrent connections impact on the system?

There is a high probability that the number of concurrent connections will affect the performance of the system. There is a high risk of faults being triggered as the number of concurrent connections increases. There are general threats that do not depend on the nature of the system.

There are various reasons for making this claim. These include:

- ! Connections from clients require network connections. These consume resources in the network software layers. The amount of processor time required to operate the network software may increase as the data structures get bigger. Caching may become less effective as more data is managed introducing further delays. There may be limits on the number of concurrent network connections that can be supported.
- ! Connections from clients require server resources. The server must hold information on the connection.
- ! A server may allocate a worker thread to a connection for the connection duration - a thread-per-connection model. There may be a limited number of workers. Exhaustion of the worker pool could cause delays to operations or rejection of transactions. Triggering of the mechanism for dealing with these could execute infrequently used functions containing undiscovered faults.

The above concerns apply whether or not the connection is active. During client processing the connection can be dormant, the server has no work to do for the connection, but it still exists as a connection. On the other hand network bandwidth saturation is not an issue. The amount of network traffic depends upon the number of active connections, not on the number of connections that exist. As the arrival rate and duration of interactions has not changed the intensity of network traffic has not changed.

### Does the reduction in the average number of concurrent transactions impact on the system?

The impact of changes in the average number of concurrent transactions was discussed in section 3.3 on page 8. The same issues arise whether the overall reduction is caused by reducing the client processing time in the simulation or reducing the operator response time.

### Does the reduction in the average number of transactions in the creating and completing states impact on the system?

The factors to consider here are similar to the ones identified for the average number of concurrent transactions in section 3.3 on page 8. Those factors were:

- ! Data structures that exist for the duration of the creation or completion activity persisting through the client processing period.
- ! Functions that select and operate on the set of transactions that are in one or both of these states.
- ! Locking of resources through the client processing period.

The effects these can have are identical to the ones described in section 3.3.

### Are changes an issue when testing?

There is a high probability that changes to the average number of concurrent connections will affect the system behaviour. The changes can affect both the performance and the integrity of the system. For systems that maintain client to server connections for longer periods there is a danger that reducing these periods will mask issues around link timeouts.

The connection mix is an issue in both Performance-Testing and Stress-Testing. Differences between the mix that the scenario would provide and the mix produced by the simulation of the scenario can make the testing invalid.

The nature of the system will determine whether changes in factors like the average number of transactions in the creating and completing states affect behaviour. Higher values may affect performance or may be a threat to the integrity of the system. If they affect performance then Performance-Testing must look carefully at the fidelity of the simulation. Similarly if they create a threat of failures then Stress-Testing needs to be certain of the acceptability of the simulation.

## 4.4 'Safe' Approaches to Eliminating Channel Idle Time in the Connected Periods

Perhaps this section should have had a question as its title - 'Is it safe to eliminate channel idle time in the connected periods?'. When the simulation has lower connection durations than the scenario would create there is a high risk of system performance changing and of faults being masked. Great care should be taken when optimising a simulation in this way. Some approaches that could help are outlined below.

### Create a number of 'dummy' connections

This approach helps to address the number of concurrent connections issue. The load generator channels are used to apply the transactions to the system using the shorter connection duration. An additional source, possibly the real client software, used to open other connections to the system. These are not active connections they are just there to create connection data structures and to tie up resources. In the example a further 80 dummy connections could be created.

The additional static connections help to improve the fidelity of the simulation but obviously it is not a complete match. The connections are static and hence the data structures are not changing much. This can improve the performance of caching and so reduce the amount of work the system has to do. The approach will help find faults during Stress-Testing such as failures due to resource exhaustion.

### Maintaining the Correct Connection Time Without Channel Idle Time

In an ideal world we would want to apply similar approaches to those used when eliminating the idle time whilst disconnected. These permitted the idle time to be eliminated without, in this example, altering transaction durations. This would mean being able to use a channel on one connection whilst it maintains others. Generally this can not be done.

### Extrapolating Test Results for Performance Testing

A testing process may do some Performance-Testing, possibly the majority, with a reduced connection life, perhaps also using dummy connections to improve the simulation. Having done this it is necessary to review the results obtained to predict what they would become if the average duration of connections had not been reduced.

The basis for this could be a comparison of resource utilisation with different numbers of connections each lasting for the predicted average duration. For example the processor usage associated with a typical connection could be estimated. When this processor overhead is known and the results of the test are available it is possible to estimate how a full implementation of the scenario would behave. This helps to compensate for the connection characteristics of the simulation.

For example suppose that the measurements shown below were taken during execution of the simulation with shortened connection time and only 24 concurrent connections.

Component	Average Measured Time
Phase-1 Connect and Read	41msec
Phase-1 Update and Disconnect	46msec
Phase-2 Connect and Read	39msec
Phase-2 Update and Disconnect	51msec
<b>Total:</b>	177msec
Processor utilisation	78%

Now if we assume the system is operating as a simple queue then the processing time for a single set of interactions can be estimated as being 39msec.

Investigation of the overhead of connections has shown each uses 0.1% of the processor time. The simulation operates with 24 connections but the scenario average is 104, there are 80 additional ones. It is estimated that these 80 additional connections will use 8% of the processor time. Now imagine there is a virtual processor, hosted on the real processor, servicing requests. The speed of this virtual processor has been reduced to 92% of its original value.

With the virtual processor speed 'reduced' the time taken to service a single request is estimated at 42.4msec. The utilisation of the virtual processor is estimated at 84.8%. Given these figures an estimated average response time of 279msec is produced.

This simple extrapolation has indicated that the additional concurrent connections predicted in the scenario but not present in the simulation could have a significant effect on the system response time. The prediction is a 58% rise over the time observed in the simulation. If the system response time were a significant component of the transaction time then further investigation would be essential.

It should be recognised that this process does not give highly accurate results. What it does do is provide an indication of how the full scenario would differ from the limited simulation. The increase in response time will almost certainly not be a 58% increase, it could be more, it could be less. What the figure does indicate is that the response time will be higher and it will not be just 5-10% higher.

## 5. Conclusion

Accelerating client / user response time is a technique that can be used to optimise the use of load generators - to allow peak loads to be simulated. One result of this approach is a reduction in the average lifetime of application domain entities such as transactions and implementation domain entities such as connections. A side effect is that the overall characteristics of the work presented to the system by the simulation are changed.

Care needs to be taken to ensure the simulation fidelity is maintained. To ensure that, for the proposed test, all significant characteristics of the simulated workload are close enough to those predicted for the scenario being simulated. If the characteristics are not close enough then the simulation must be adjusted or a method of adjusting the results to compensate for the differences needs to be devised.

A failure to identify that a characteristic of the scenario is not accurately simulated can lead to ineffective testing. The results of the tests may not represent what will actually happen when the system is live and encounters the situation that is represented by the scenario. Ultimately this can mean that live systems display unforeseen performance and reliability problems.

## Author Biography

The author of this paper is Neil Hudson, the principal consultant of SQC Technology Ltd. His career has focussed on software assurance, it has encompassed software testing, software test management and software test automation. He also has experience of software development and of managing development projects. A Chartered Engineer and a British Computer Society Registered Consultant he holds qualifications in software testing, project management and high integrity systems development.

Neil has 15 years of industrial experience in the software quality assurance arena. He has been responsible for management and delivery of assurance activities on a wide variety of systems. Technical activities undertaken include detailed design analysis, component/integration/system testing and the design development & operation of effective automated test solutions. Managerial activities undertaken include process formulation, the definition of test strategies and the management of test teams.

He has led the evaluation & testing of complex technical software systems including large distributed systems, real-time control systems and communication network systems. During this time a number of approaches and techniques for use in software quality assurance have been developed. These include novel approaches to test planning and management, rigorous test design techniques and a original structured approach to test automation.

To discuss the ideas contained in this note please contact Neil by email at [nahudson@sqc.co.uk](mailto:nahudson@sqc.co.uk). Further information on approaches to software testing and test automation can be found at [www.sqc.co.uk](http://www.sqc.co.uk)

## SQC Technology

SQC Technology provides services and consultancy in the fields of software testing, software test management and software test automation. These include formulation of software test strategies, analysis and development of test suites, test automation and interim management. More information can be found on the SQC web-site at [www.sqc.co.uk](http://www.sqc.co.uk)