Report Paper:

MatLab/Database Connectivity

Samuel Moyle

March 2003

Experiment Introduction

This experiment was run following a visit to the University of Queensland, where a simulation engine has been created using *MatLab* SimuLink. This simulator is developed for use as the underlying engine in future User Interface (UI) experiments. The simulator engine currently gets/puts information via a DDL link to an *MS Excel* spreadsheet. While this is suitable for initial testing, when UI experiments are run the data transfer method will need to be faster. There is a need for more data to be collected in order to accurately assess system state at any time during the experiment.

There are two methods currently proposed to achieve this latter goal. One is to send a record to the database when the state of some component changes. The other is to capture the entire system state at pre-determined times then save the entire system state. The former method is expected to have less impact on resources, but prevents making temporal comparisons (unless you wish to replay the entire experiment). If possible, the latter method is preferred.

It was recognised that some bottlenecks exist under the current system design. Notably, the simulation engine requires significant processor power to run effectively and that the resources required by *MS Excel* slow processes. The experiment has the following goals:

Is a single, large, transaction more efficient than multiple smaller transactions?

Does machine speed/specifications have an effect on how the experiment runs?

Does the ODBC processing overhead have an adverse impact on data transfer speed?

Does network connectivity speed have an impact on data transfer speed?

Does the number of fields being transferred have an impact on data transfer speed?

By identifying what combination is most effective we can then determine an optimal hardware setting for future UI experiments.

This experiment is comparative in nature. It is not fully robust due to limitations in tool availability. However, the comparisons made may be useful in eliminating some of the possibilities and guiding further experiments.

Three hardware setups were used. Two computers are similar in specification, but with different network connection speeds. The third was a much lower specified, but served to provide a comparative role – how much difference does improved machine specification make?

Method

The first series of automated tasks was run from within *MatLab*. An array is created comprising 1000 records, each with 4 fields. This array is populated with random numbers. The array is the written:

1. From *MatLab* to a local *MS Access* database
2. From *MatLab* to an *SQL Server 7* Database

When written, each record is automatically allocated a unique identifier and time-stamped.

This series is repeated, only this time each record consists 253 fields. This value is

chosen due to a limitation of *MS Access*– each record may consist no more than 255 fields. The additional fields are the unique identifier and time-stamp fields. When complete, *MatLab* is closed.

The second automated task is run from *MS Access*. The data values in the *SQL Server 7* are transferred to tables within *MS Access*, so they can be used for comparisons later. The tables are then emptied, and data is transferred:

3. From the *MS Access* databases to the *SQL Server 7* database

The results of this final data transfer are also recorded within *MS Access* for later analysis.

The experiment is repeated under changed hardware circumstances as follows:

[P3, 10] PIII 733mhz, 512mb RAM, 10mb/s LAN connection

[P4, 10] P4 2.4ghz, 1Gb RAM, 10mb/s LAN connection

[P4, 100] P4 2.4ghz, 768mb RAM, 100mb/s LAN connection

This combination isolates computer specification, network connection speed and *MatLab* overhead (hence the *MS Access* to *SQL Server 7* run). The results from each run were stored in separate *MS Access* databases, and results drawn using identical SQL queries. This is not a factorial experiment, but a comparative effort.

Hardware

The experiment was run using 3 different computers:

- The first was a PIII 733, equipped with 512MB of SDRAM, and connected to *SQL Server 7* via a 10Mb/s LAN connection.
- The second machine was a P4 2.4, equipped with 1GB of DDRAM, using the same 10Mb/s LAN connection as the PIII machine.
- The third machine used was a P4 2.4, equipped with 768MB DDRAM, with a 100Mb/s LAN connection to *SQL Server 7*.

Software

All computers used were set up with *MS Windows 2000*, *MS Office 2000 Professional* and *MatLab* 6.1.0.450 revision 12.1. Wherever possible, background processes were either removed or closed to ensure each machine was running in as close to identical a manner as possible.

Method

Data was created within *MatLab*, formed by creating a matrix with a series of random number values (stored as type double). The time taken to create data was not included for the purpose of this experiment.

Data was then transferred from *MatLab* to *MS Access* using the database connectivity (ODBC) tools as provided by the *MatLab* Database Toolbox. The data was then transferred from *MatLab* across the LAN to the *SQL Server 7*. This process is repeated for matrices with 4 columns per row then 253 columns per row. Each matrix contained 1000 rows.

Once the *MatLab* process was complete, *MatLab* was closed and *MS Access* opened. A process was then run that gathered the timestamp information for each row written

to the *MS Access* tables and the *SQL Server 7* tables. The *SQL Server 7* tables were then emptied, and the row data in *MS Access* was written to *SQL Server 7*. The timestamps were again collected, and the *SQL Server 7* tables were emptied ready for the next iteration.

The time-stamp information is used to generate an average 'transactions per second' figure that can be compared between computer configurations and run. Preliminary tests indicated that the results between runs were not dissimilar (the averages over three runs were identical within 3 decimal places), and that a single run would be a fair representation. The processes (within *MatLab* and *MS Access*) were automated so that the process was identical in every respect.

Results

ODBC vs. MatLab

One possible bottleneck when getting data from *MatLab* is the ODBC connection. By comparing the data transfer rate of identical record sets between *MatLab* and *MS Access*, *MatLab* and *SQL Server 7* against *MS Access* and *SQL Server 7*, it was hoped that we could identify if this was the case.

When writing records with 4 fields from *MatLab* to the local *MS Access* database, the average was 51.26 transactions per second. When writing the same records from *MS Access* to *SQL Server 7*, the average was 386 records per second (about 7.5 times faster). This suggests that *MatLab* was causing the bottleneck, rather than the ODBC link. This ratio of difference averages 602% across the series, as shown in the following table:

	P3, 10	*P4, 10*	*P4, 100*	*Averages:*
MatLab/MS Access (4)	51.26	130.14	113.63	98.34
MS Access/SQL (4)	386	436	815	545.67
	753%	**335%**	**717%**	**602%**

Machine Speed and Specifications

For all experiment components involving *MatLab*, there was a significant performance increase when using the higher speed CPU machines. In pre-experiment testing, the MS Windows 2000 Resource Monitor showed that, when *MatLab* processes were running, 100% of available CPU resource was being used. This was true of all machines used in testing.

The table below shows that greater than 50% performance improvements were made with the greater CPU speed.

	P3, 10	*P4, 10*	*P4, 100*	*P3, 10 / P4, 10*	*P3, 10 / P4, 100*
MatLab/MS Access (4)	51.26	130.14	113.63	**254%**	**222%**
MatLab/MS Access (253)	2.60	7.20	7.14	**277%**	**274%**
MatLab/SQL (4)	54.82	116.63	112.00	**213%**	**204%**
MatLab/SQL (253)	2.38	4.63	4.89	**195%**	**206%**
MS Access/SQL (4)	386.00	436.00	815.00	**113%**	**211%**
MS Access/SQL (253)	41.96	48.95	148.33	**117%**	**354%**

Interestingly, the machine with the greater memory (P4, 10) performed better than the similar machine (P4, 100) with greater LAN connection speed in all components

involving *MatLab*. This suggests that *MatLab* is both CPU and memory hungry. This difference could be explored more fully in later experiments.

Improved machine performance is also an important factor when comparing results of data transfer between *MS Access* and *SQL Server 7*. The ratio of difference is lesser than the gains for *MatLab*. The crossover between network speed and machine specifications appears to be transferring large files across the network. Note that *MatLab* requires significantly more resource to operate than *MS Access*, so the positive effect of improved machine specification is expected to be more significant than network speed.

Network Speed

The best comparison for the issue of network speed was to compare similar machines, and their data transfer rate between *MatLab* and *SQL Server 7* and between *MS Access* and *SQL Server 7*. Values are as shown in the following table:

	P4, 10	*P4, 100*	% increase
MatLab/SQL (4)	116.63	112.00	**104%**
MS Access/SQL (4)	436.00	815.00	**187%**

One factor not accounted for in this comparison is the volume of other network traffic. Although the differences are not significant here they may be more significant if this was a closed network (that is, only these machines being connected). Theoretically network traffic from other sources should not be a major influencing factor in this study as times for running the experiment were deliberately chosen at times of low overall network traffic.

However, the results confirm that where there is a significant processing overhead (as with *MatLab*) network speed is less of an issue that machine resource.

Records per Transaction

One of the main objects of this exercise was to establish what sort of record keeping was possible from the simulation engine to data store, reflecting changes in systems state. Those changes in state can then be assessed with regard to time. For example, in proposed UI experiments, there may be state changes caused by outside influence (an experiment controller causes a component to fail), the system itself (flow on effects as a result of controlling the system), or by the experiment participant (attempting to control the process). Being able to replicate an entire state according to time makes later comparison significantly easier. Also, a time segment can be specified and replayed easily, based on time snapshots. Reaction time and desirability of reaction can be calculated given the time of events then plotting the expected/desired/actual outcome.

These system state changes may be recorded individually, and recorded to a table. This would be fast (small number of field values), but this experiment reveals that this is slower than a single large transaction due to the sheer volume of transactions. Instead, an option would be to bundle the entire system state on a regular basis (say once per second) and send this as one record. The aim in practice is to find a balance between a large record and multiple smaller records.

In the following table, the larger the value reflects better comparative efficiency (smaller transaction over larger transactions):

	P3, 10	*P4, 10*	*P4, 100*
MatLab/MS Access (4)	51.26	130.14	113.63
MatLab/MS Access (253)	2.60	7.20	7.14
	31.17%	**28.57%**	**25.17%**
MatLab/SQL (4)	54.82	116.63	112.00
MatLab/SQL (253)	2.38	4.63	4.89
	36.48%	**39.84%**	**36.20%**
MS Access/SQL (4)	386.00	436.00	815.00
MS Access/SQL (253)	41.96	48.95	148.33
	14.55%	**14.08%**	**8.69%**

This would suggest that, when sending data from *MatLab* to *SQL Server 7*, there are greater efficiencies when sending a single large record over many smaller records. Also, straight comparisons would suggest that sending to *MS Access* locally is faster than to *SQL Server 7* across the network. This despite the significant resource hit to be had while running applications simultaneously.

Conclusions

Running the simulation engine on a computer featuring as fast a processor and as much memory as possible seems to be the key. Although a fast network connection is desirable, it is not necessarily going to result in as significant a speed impact.

The database connectivity standard of choice would be ODBC. This does not provide any significant performance difference between application or platform. The choice of DBMS does not seem to make any significant difference to performance. Given the ready availability of *MS Access*, the option of running this on a separate machine would be a simple, cost-effective, option. Bearing in mind the field number limitation, it may be necessary to run several tables rather than a single large table (as would be available in *SQL Server 7*). There is no requirement to investigate this further at this time.

Network connectivity is less important than machine specification. If using an independent network the DBMS server would not have to be hugely fast, but as always faster is better.

The hypothesis that multiple, small, transactions is more efficient than larger, more complete, transactions, was found to be incorrect. The smallest efficiency rating in favour of large transactions was 8.7%, the largest 39.84%. Future experiments may include both large and small record sizes, but the efficiency of using larger transactions is clear from this comparative work.

The hypothesis that machine specification would affect results in a positive manner was confirmed. Surprisingly, [P4, 10] was found to provide better results than [P4, 100] in some instances. This needs to be investigated further.

It was hypothesized that the major overhead would be MatLab processes rather than ODBC. This is confirmed when comparing results of data transfer between MatLab and SQL Server against MS Access to SQL Server. In each configuration, MatLab was considerably slower than MS Access. This suggests that MatLab processes are indeed highly resource hungry. Future experiments may concentrate on comparing

specification differences on well-specified machines, without having the need to compare these against older, slower, machines.

Does network connectivity speed have an impact on data transfer speed? This experiment suggests that there is some improvement when faster network speeds are involved. However, we cannot say this with certainty as the best source of comparison was tainted by different memory specification between otherwise identical machines. This then should be investigated further.

Future Work

The ability to run different *MatLab* processes in multiple threads (either AMD, P4 with hyper-threading, or multiple processor environments) may result in further timesavings. An alternative would be to host the *MatLab* simulation engine on the same machine as the DBMS, but explicitly use separate processors and OS threads. These options were not tested in this experiment.

It would be useful to follow this with a more complete experiment comparing 2 memory configurations in a single high-speed machine, then test across both slow and fast network conditions. It was clearly evident that greater resource resulted in performance improvements, and that the memory issue may have contributed to this performance improvement. A factorial experiment may be run to clearly quantify how much effect memory or network speed has on overall performance.

Acknowledgements

I must thank Brendon Sly for making *SQL Server 7* resources available, and for ensuring that all software was installed in a consistent manner across the differing machines used.

Also, thanks to Rizah Memisevic for his enthusiastic simulator descriptions and assistance during my visit to the University of Queensland.

Appendix I – Experiment Aims and Method

Aim: test database connectivity between *MatLab* and an ODBC data source for the purpose of writing large datasets in large quantities (i.e. many large records, many small records).

This was tested to ensure that the following confounding factors were eliminated, also ensuring that, wherever possible, bottlenecks are identified and documented.

The tests are run to confirm the amount of data that can be passed to a database table from *MatLab* in a timely fashion. The aim is to pass the system state of a simulator to a database at regular intervals (preferably <1s). This is reportedly dependant upon the following factors:

- Machine speed (how fast *MatLab* will run)
- ODBC processing overheads
- Machine connection speed
- The number of field to be written in any given transaction

It is proposed that an experiment be run to test these factors, identifying which potential bottlenecks may be allowed for later.

The experiment has been structured as follows:

Create an array comprising 1000 records, with 4 fields. This array is populated with random numbers. The array is the written:

- From *MatLab* to a local *MS Access* database
- From *MatLab* to an *SQL Server 7* Database
- From the *MS Access* database to the *SQL Server 7* database

When written, each record is automatically allocated a unique identifier and time-stamped.

This series is repeated, only this time each record consists 253 fields. This is chosen due to limitation of *MS Access*– each record may consist no more than 255 fields. The additional fields are the unique identifier and time-stamp fields.

The experiment is repeated under changed hardware circumstances as follows:

Run 1 PIII 733mhz, 512mb RAM, 10mb/s LAN connection

Run 2 P4 2.4ghz, 1Gb RAM, 10mb/s LAN connection

Run 3 P4 2.4ghz, 768mb RAM, 100mb/s LAN connection

This combination effectively compares CPU Speed, network connection speed and *MatLab* overhead (hence the *MS Access* to *SQL Server 7* run). The results from each run will be stored in separate *MS Access* databases, and results drawn using identical SQL queries.

The time-stamp values are used to calculate average record write times per second, therefore determining which combination is quickest/requires least resource.

All software (*MS Access* 2000, *SQL Server 7*, *MatLab* 6.1) is identical across each machine. Where possible, all background processes have been eliminated. All experiments are run across the University of Otago Novell Network.

One other permutation is possible, but will only be run where possible. That is where *SQL Server 7* and *MatLab* may be run concurrently on a single machine.

Appendix II – Numerical Results

The following table is linked to a *MS Excel* spreadsheet. This represents the raw values and corrected values for each experiment component.

The corrected values are the raw values less the beginning and end seconds. This allows for the possible skewing effect that may be caused through a lesser number of transactions during the beginning and end seconds.

	P3, 10		P4, 10		P4, 100	
	raw	corrected	raw	corrected	raw	corrected
MatLab/MS Access (4)	47.61905	51.26316	111.1111	130.1429	100	113.625
MatLab/MS Access (253)	2.597403	2.600522	7.142857	7.202899	7.092199	7.136691
MatLab/SQL (4)	52.63158	54.82353	100	116.625	100	112
MatLab/SQL (253)	2.375297	2.37619	4.608295	4.627907	4.854369	4.892157
MS Access/SQL (4)	250	257.3333	250	290.6667	500	500
MS Access/SQL (253)	76.92308	41.95652	47.61905	48.94737	125	148.3333

There is one exception – when the 4 field records are written from *MS Access* to *SQL Server 7*. These transactions took place within 2 seconds – if the begin and end seconds are removed then all transactions would be lost. As a result, Values are to the nearest millisecond (using an SQL query).

The following graph shows the corrected values by machine and experiment component.