Public Sector Scotland Shared Support Services Project

Reliable 10gAS-Forms-Services-Finance Version 3.2.1B

Performance Trial

Report

David Edwards
Principal Consultant
E2 Systems Limited
11th September 2008

Table of Contents

Management Summary

The Public Sector Scotland Shared Support Services Project (Public Sector) seeks to consolidate the Financial functions for a number of existing Public Sector Bodies into a single Shared Support Service.

Reliable is offering its 10gAS-Forms-Services-Finance Product to meet this need, deployed on an ORACLE Real Application Cluster (RAC).

10gAS-Forms-Services-Finance follows an industry-standard Three Tier Architecture.

- The End User accesses the application from within their Web Browser on their PC.
- The PC communicates with an Application Server Layer
- This in turn communicates with a Database Server Layer.

Reliable have proposed that the Application Server Layer be delivered as a series of cluster-capable Applications Servers, and that the Database Layer be delivered as a series of connected Database Servers sharing access to a single database.

The beauty of this solution is that it can readily be expanded to deal with more users and more usage by adding extra machines to each layer, with minimal impact on the machines that are already in situ. This so-called Horizontal growth strategy should minimise the Total Cost of Ownership of the system going forward.

E2 Systems, the United Kingdom's leading independent Performance Testing Specialists, were retained by Reliable to demonstrate that:

- 10gAS-Forms-Services-Finance can support upwards of 2,000 concurrent users
- Demonstrate that 10gAS-Forms-Services-Finance is stable under load and degrades gracefully as additional users come online
- Accessing a database scaled up from the existing Greater Glasgow database so that it had 3 years of history for the whole Public Sector user base
- Running on a full size scale production-ready hardware solution conforming to the above architecture, provided by a Manufacturer at their Benchmark Centre.

Briefly, the demonstration was successful.

- The End User Response Times satisfied the Technical Success Criteria.
- The Break test demonstrated that 10gAS-Forms-Services-Finance performance degrades gracefully as the number of concurrent users increases
- The Soak test demonstrated that 10gAS-Forms-Services-Finance remains stable under continuous load
- The system throughputs exceeded the levels required for the system to get through its on-line work in the on-line day.

Sufficient kit to support 2,000 concurrent users was put together over a weekend. As our testing progressed we encountered a number of issues that

- Disrupted our schedule
- Could be worked round, or simply ignored, as we strove to demonstrate that 10gAS-Forms-Services-Finance met the Technical Success Criteria.
- But which would need to be resolved if they ever recurred as live running approached.

Since we believe that they are likely to be of interest to Public Sector and Partner, we detail these in section 3.2.

We also discovered a number of opportunities to improve the performance of 10gAS-Forms-Services-Finance still further. Reliable has accepted these points, and has worked them in to the development schedule for 10gAS-Forms-Services-Finance version 3.3.S1. A list of the opportunities, and Reliable’s specific responses to them, appears in a table in section 4.2.

However, to re-iterate, even though we discovered opportunities for improvement.

- A system comparable to that proposed for Public Sector was put together and commissioned
- The system throughputs exceeded the levels required for the system to get through its on-line work in the on-line day.
- The End User Response Times satisfied the Technical Success Criteria.

Thus, we have no hesitation in asserting that the Performance Test establishes the viability of the proposed, cost-effective, horizontally scalable hardware solution. Full details may be found in the body of the report.

1. Introduction

The Public Sector Scotland Shared Services Project (Public Sector) seeks to consolidate the Financial functions for a number of separate Public Sector organisations into a single Shared Service.

Reliable is offering its 10gAS-Forms-Services-Finance product to meet this need, deployed on an ORACLE Real Application Cluster (RAC).

10gAS-Forms-Services-Finance follows a standard Three Tier Architecture.

- The End User accesses the application from with their Web Browser on their PC.
- The PC communicates with an Application Server Layer
- This in turn communicates with a Database Server Layer.

Reliable have proposed that the Application Server Layer be delivered as a series of independent Applications Servers, and that the Database Layer be delivered as a series of connected Database Servers sharing access to a single database

The beauty of this solution is that it can readily be expanded to deal with more users and more usage by adding extra machines to each layer, with minimal impact on the machines that are already in situ; a so-called Horizontal growth strategy.

Public Sector will have many more users, and many more transactions, than any current 10gAS-Forms-Services-Finance installation.

Reliable has not previously been deployed on an ORACLE RAC.

There is thus a requirement to demonstrate that 10gAS-Forms-Services-Finance can support upwards of 2,000 concurrent users with the proposed hardware solution.

E2 Systems, the United Kingdom's leading independent Performance Testing Specialists, were retained by Reliable to demonstrate that:

- 10gAS-Forms-Services-Finance can support upwards of 2,000 concurrent users
- Accessing a database scaled up from the existing Greater Glasgow database that had 3 years of history for the whole Public Sector
- Running on the full size scale production-ready hardware solution conforming to the above architecture, provided at a Manufacturer's Benchmark Centre.

This document records the outcome of the exercise. It is addressed to the Public Sector Project Team, and Partner management who would have operational responsibility for the system.

This report is organised into sections. This Introduction is preceded by a Management Summary, and is followed by:

- The Method and Approach, (section 2), which follows the format of the originally proposed plan of action.
- The Findings (section 3), which presents the qualitative and quantitative results of the Performance Trials.
- The Conclusions (section 4) that we draw with respect to the likely behaviour of an 10gAS-Forms-Services-Finance Public Sector.

Details concerning the scripts run (Appendix A) and the hardware configuration tested (Appendix B) are to be found in Appendices.

Whilst 10gAS-Forms-Services-Finance is built around an industry standard three-tier architecture and should take advantage of most Oracle and operating system optimisations transparently, the ancestors of 10gAS-Forms-Services-Finance Version 3.2.1B were hand-optimised for historical ORACLE releases from ORACLE Version 5 onwards, and current 10gAS-Forms-Services-Finance users being subsumed within Public Sector have had performance issues with 10gAS-Forms-Services-Finance, so it would scarcely be credible if a testing exercise did not uncover performance improvement opportunities, and indeed some were found.

A table of these opportunities, together with the Reliable response, can be found in section 4.2.

The exercise was carried out in a compressed time-scale. We would like to thank all concerned for their forbearance and co-operation whilst we carried out our parts of this process.

2. Method and Approach

2.1 Introduction

A successful Performance Trial Process naturally follows if we

- Remain focused on the objectives.
- Work with a large enough benchmark to adequately model the live system.
- Restrict the functionality to be included so as to limit the build phase to fit the available time.
- Make maximum use of the benchmark environment once it has been built.
- Utilise a restricted time span to best effect.

The purpose of the load test is to confirm that the Reliable 10gAS-Forms-Services-Finance FMS can scale to handle 2,000 concurrent users – the projected size of the combined user population of the shared service for Public Sector Scotland.

The test should formally confirm the product's ability to scale to the required volumes – both concurrent users and transactions.

We followed a plan that satisfied our objectives, and ensured that Reliable would be given the opportunity to address application performance issues that emerged from the exercise, if there were any, in advance of the publication of the report.

The plan was conceived as a series of phases. These were:

- Confirm Objectives and Agree Scenario
- Build Test Database
- Create Scripts
- Carry out 'Sighter' runs to tune the system
- Run Formal Benchmark
- Report the results.

Each phase depended on its predecessor. There may in principle be slack time between each phase, but the business timing constraints precluded it. In the event we took about 6 weeks elapsed.

2.2 Confirm Method and Objectives

The precise nature of the workload to be tested (number of users carrying out which application tasks), and any variant scenarios to be tested, has first to be agreed. Not having the benefit of program usage statistics gathered from a live system, we used the following heuristics, based on our experience of many systems gathered over many years, to arrive at a projected workload.

The bulk of data in the system will be entered in 'core' time, of which we assume there will be 120 hours each month.

We should test a busy hour representing 3 times the long term average.

If we were to measure the amount of time spent by 10gAS-Forms-Services-Finance users using the various 10gAS-Forms-Services-Finance facilities, we would find the most frequently used 9 functions account for 75 percent of all 10gAS-Forms-Services-Finance activity, so we need to uplift our chosen transactions by 1/3 to account for all the other activity.

We multiply the Greater Glasgow numbers by 2000 (= Projected Maximum Concurrent Usage of Public Sector) /370 (= Observed Maximum Concurrency at Greater Glasgow) = 5.4 to get the Shared Service total.

By combining these factors we end up dividing the monthly totals of Greater Glasgow transactions by 5.5 to come up with our required busy hour transaction throughputs.

This gives us, to start with, 2700 Provision Entries (STK080), each with 7 lines, (5 stock items, 2 purchase items).

This accounts for only half of the Purchase Orders; the others are input directly.

Thus we get 1350 Purchase Orders input (POP005), with 4 lines each.

Goods Receipts and Purchase Invoices must match the Purchase Orders, both those deriving from the Provisioning and those entered directly, so we get:

2700 Goods Receipts input (POP009), with 4 lines each.

2700 Purchase Invoices input with ACT35O, with 4 lines each, marked for posting with ACT091.

2700 Stock Issues Picks (STK081) of 5 lines each.

1250 Stock Take Results (STK054) followed by Stock Balance Adjustment (STKR56).

Finally, we assume that someone actually using the system will go through 10 business transaction cycles an hour. This would be 10 Purchase Orders Entered, 10 Purchase Invoices Entered, 10 GL Drill Down cycles, or whatever. This gives us the number of users performing each transaction.

The residual users are assigned to Enquiries.

Certain reports and batch processes form a necessary part of the business processes, and these should be run appropriately during the 'Busy Hour'.

If this doesn't produce enough report requests to account for the observed frequency of report requests at Greater Glasgow, uplifted as above to model Public Sector, a selection of other reports would be run at a frequency designed to ensure that the observed frequency of report execution at Greater Glasgow, scaled up for the larger number of users, is matched.

The following table shows the number of update users of each type, and the total number of transactions that they must aim to complete between them in an hour.

Screen	**Description**	**Users**	**Total Cycles/hour**
STK080	Provision Entry, each with 7 lines, (5 stock items, 2 purchase items).	270	2700
STK081	Provision Pick Entry, 5 lines each	270	2700
POP005	Purchase Order Entry, 4 lines each	135	1350
POP009	Goods Receipt Entry, 4 lines each	270	2700
ACT35O	Purchase Invoice Entry, 4 lines each	270	2700
STK054	Stock Take Results Entry, 1	125	1250

Screen	Description	Users	Total Cycles/hour
	stock item		
Sub-total	Update Users	1340	

The following table shows projected activity levels for the Enquiry users, and gives the user totals.

Screen	Description	Users	Cycles
AC101P	Supplier Enquiry	100	1000
STK082	Provision Enquiry	80	800
ACT041	GL Drill Down – Bottom Level Cost Centre, Second Level Account, YTD Period 12	300	3000
ACT041	GL Drill Down – Fourth Level Cost Centre, Second Level Account, descending, YTD Period 12	80	800
ACT041	GL Drill Down – Second Level Cost Centre, Second Level Account, descending, YTD, Period 12	19	190
ACT041	GL Drill Down – Top Level Cost Centre, Second Level Account, YTD Period 12, descending	1	10
POP016	Universal Enquiry	80	800
Sub-Total	Enquiry Users	660	
Grand Total	Update Users + Enquiry Users	2000	

The ACTS041 Enquiry Profile is as requested by Public Sector. We believe that it may be rather onerous. On the expanded database, a single Top Level ACT041enquiry summarises perhaps 1,500,000 balances.

The technical success criteria were defined as follows.

Transaction Type	Response 70 percentile better	Examples
Simple single target enquiry, simple insert	2 seconds	Apply a purchase order line, look up transactions for a supplier
Inserts and updates of medium complexity	5 seconds	Confirm a Purchase Order, Confirm a Goods Receipt
Complex enquires	10 seconds	Retrieve Second Level Account/Bottom Level Cost Centre in ACTS041

The following table shows how many times the reports and batch processes that form part of the normal Business Processes were executed in the first four months of live running at Great Glasgow, together with statistics for ACT_RG3, the most commonly executed Management Report.

Report	Requests	Running	Percentage	Percentage	Description
pop_r05	50324	50324	37.6187	37.6187	Purchase order print
pop_r56	21750	72074	16.2588	53.8774	Print internal delivery notes
stk_r71	13555	85629	10.1328	64.0102	Provision delivery note

Report	Requests	Running	Percentage	Percentage	Description
pop_r77	9683	95312	7.23833	71.2485	Shelf stacking report
act_rf5	5605	100917	4.1899	75.4384	Invoice batch posting
stk_r72	4799	105716	3.58739	79.0258	Provision replenishment
act_rg3	4044	109760	3.02301	82.0488	WTE budget; period report
req_r22	3071	112831	2.29566	84.3445	Automatic purchase order creation
pop_r09	2072	114903	1.54888	85.8934	Goods receipt and return note print
stk_r62	1265	119722	0.945625	89.4957	Stock requisition print
stk_r74	1092	120814	0.816302	90.312	Provision list
stk_r09	820	125536	0.612974	93.8419	Auto-replenishment of stock routine
stk_r70	113	132231	0.0844708	98.8466	Provision picking list
stk_r56	59	132917	0.0441042	99.3594	Update stock take results into stock balances routine
pop_r08	49	133068	0.0366289	99.4722	Goods received not yet invoiced report

The number of report invocations expected in a busy hour, following the logic used to arrive at the levels of updates, would be 6150.

It can be seen, as may be expected, that any Purchase Order entered is also printed. However, in the terms of our scenario, if we aimed to ensure that all Purchase Orders, Goods Receipts, etc., that are entered in the busy hour, are also printed in the same hour, the number 6,150 would be far exceeded.

In addition to the Business Process reports, Public Sector explicitly requested that we would run

STKR06 Stock Valuation
STKR24 Stock Summary
ACTR96 Refresh Commitments Routine

Our strategy was to use historical report requests for all of the reports in order to produce 'model' report requests, produce report submission commands conforming to these models that targeted the data that our scripts were adding to the database, randomise the order of these requests, divide them into equal blocks for the Application Servers (see below), and see how many would be processed..

2.3 Build Database

The starting point was a scrambled snapshot of the Greater Glasgow database, taken when it had 3.67 periods of live transaction data loaded.

By comparison, the Public Sector Shared Service, after three years will have:

- 23 Business Units
- 5.4 times as many users
- 5.4 times as much data (Public Sector Greater Glasgow is one of the largest)
- 36 periods in total

We observe that 5.4 times 36/ 3.67 (= 52) is approximately 6 * 36/4 (=54)

So if we treat our 3.67 periods as 4, and use a factor of 6 for the business complexity, we end up with slightly more records that we need, and we simplify the scaling approach greatly.

Our scaled up database was built using the following principles.

- 6 times as many sub-ledgers
- 6 times as many cost centres
- 6 times as many users

We assumed that the chart of accounts is standardised, and therefore we need not increase the numbers of records in ACCOUNTS and related tables.

We assumed that Greater Glasgow, as the largest single entity joining the shared service, already has most of the possible suppliers loaded.

We were not testing Sales Ledger functionality.

Therefore, we did not add further NAMES.

We added 6 times as many products, since these are set up once per warehouse, and there are 6 times as many warehouses.

Most of the JOBS entries are Fixed Assets, whose functionality wasn't being tested, so we left JOBS unchanged.

There are hardly any ACTIVITIES, so we left them, likewise, unchanged.

Transaction data from the first 4 periods of this year was cloned twice for this year and 3 times for each of 2 further years to give us the full size database.

Potentially large tables in application areas that were not scripted were left as they were, to save space. Chiefly this affects AUDITS, and Invoice Writer tables. In contrast, payment-associated tables were scaled, even though we did not script payment processing, because they may be the targets of supplier enquiries.

The scrambled data from Greater Glasgow was in flat files, accompanied by SQL*Loader scripts.

The database was populated as follows.

The standard 10gAS-Forms-Services-Finance Service Creation approach set up the schema, and populated tables with the minimum data for an initial 10gAS-Forms-Services-Finance implementation.

Tables with no rows at Greater Glasgow, or which were specifically excluded (eg. Budget-related data) continued to have no rows. But note that the table used for budget versus actual enquiries, ACCOUNT_BALANCES, was present.

Tables that did not need any more data than was present at Greater Glasgow were loaded from the flat files.using SQL*Loader.

For the other tables, the scrambled data from Greater Glasgow was passed through a filter process.

This added 5 extra output records for each input record to account for the increase in the number of sub-ledgers and cost centres.

In the case of transaction records that fell in the first 4 periods of the current year, 8 further output records (one for each of the other 4 month periods needed to take us up to 3 years worth of data) were added for the 6 output records that result from the sub-ledger/cost centre scaling.

Applying this process consistently to all the key values in each record ensured that the integrity of the database was preserved.

To make this clear, we give below some detailed examples of how this worked in practice.

Each existing cost centre was allocated a sequential number, between 1 and about 24000.

5 alphabetic characters that were observed to not be used as the first character of an existing cost centre were identified.

When a cost centre was encountered:

- Its number was looked up
- The five cloned records have, for their cost centre, each letter in turn followed by this number in hexadecimal.

Sub-ledgers are handled analogously.

Key values associated with sub-ledgers were simply cloned by using the extra sub-ledgers.

To get values for the extra periods, a number greater than the number of values in the 4 month period was added to the sequence for each duplicate.

For some of the columns that are being expanded as part of the on-going 10gAS-Forms-Services-Finance development, the letter plus number technique overflowed the current column length. In these cases, UUENCODING was used on the binary representation of the looked up number to produce a compact, printable representation compatible with input field format rules that coerce user input to upper case.

Below is a table detailing the row populations that resulted.

Table	Rows	Table	Rows	Table	Rows
ACCOUNTS	311	ACCOUNT_BALANCES	3452080	ACCOUNT_FOR_ASSET_TYPES	30
ACCOUNT_HIERARCHIES	1310	ACCOUNT_STRUCTURES	5	ACTIVITIES	3
ACTIVITY_HIERARCHIES	3	ADVICE_REGISTER	465680	ALLOCATIONS	777406
ARGUMENT_HEAD	13378	ARGUMENT_ITEM	95500	ASSET_LOCATIONS	58
AUDITS	2383080	AUDIT_EXCEPTION_TABLES	2	AUDIT_GROUPS	4
BATCHES	33036	BR_CASHBOOKS	18	BUDGET_TRANSACTION_SUB_TYPES	9
CASH_BATCHES	9	CASH_ON_ACCOUNT	6	Reliable_DISALLOWED_PASSWORDS	0
Reliable_PASSWORD_AUDIT	0	Reliable_PASSWORD_HISTORY	0	Reliable_PASSWORD_PROFILES	0
CFACSPLUS	756	CFACSPLUS_TEXT	175	CFACS_ALL_ADMIN_DATA	6
CFACS_ALL_LOGIN_COMMANDS	2	CFACS_ALL_SERVICE_DATA	6	CFACS_ALL_USER_DATA	1151
CFACS_GROUP_ITEMS	240	CFACS_LOADER_COLUMNS	103	CFACS_LOADER_PROGRAMS	7
CFACS_SEQUENCES	2	CHEQUE_MARK_NUMBERS	6	CIT_CODES	1
COMMITMENT_TRANSACTIONS_AUDIT	35218	CONSOLIDATION_METHODS	5	CONTRACT_PRICES	1669
CONVERSION_FACTORS	173	COST_CENTRES	14106	COST_CENTRE_HIERARCHIES	36294
COST_CENTRE_STRUCTURES	1	CREDIT_CODES	1	CREDIT_CONTROLLERS	2
CROSS_RATES_HISTORY	2	CUT_OFF_STOCK	3939	DD_MANDATES	11
DELIVERY_SCHEDULES	8964	DETAIL_PROPERTIES	5	DISPUTE_AUDITS	1209
DISPUTE_REASONS	5	DRILL_DOWN_DICTIONARY	6	EANALYSER_QUERY_DETAILS	0
EANALYSER_SAVED_QUERIES	17	FAT_STATUS_CODES	1	FRAMEWORK_USER_APPLICATIONS	16

Table	Rows	Table	Rows	Table	Rows
GL_COMMITMENTS	3023848	GL_TRANSACTIONS	19829994	GOODS_RECEIPTS	359651
GRN_LINES	794245	GROUP_INFORMATION	4	HEADER_INFORMATION	4
HEADER_PROPERTIES	3	INTEREST_RATES	2	INTERNAL_DELIVERY_ADDR ESSES	5071
INTERNAL_SEQUENCES	3	INVOICE_ANALYSIS	2153568	INVOICE_LINES	1244955
INVOICE_NUMBERS_ONE	2	IW_INVOICE_HISTORY	0	IW_INVOICE_LINES	743
IW_INVOICE_LINE_TEXT	3696	IW_INVOICE_TEXT_LINES	892	IW_PRODUCTS	26
IW_PRODUCT_ANALYSIS	1	IW_PRODUCT_TEXT	62	IW_TEXT_DEFAULTS	4
IW_TEXT_DEFAULT_LINES	4	JOBS	77	JOB_HIERARCHIES	77
JOB_TYPES	2	JOURNALS	30422	JOURNALS_INTERNAL	1932454
JOURNAL_LINES	20755796	JOURNAL_SUB_TYPES	2	JOURNAL_TYPES	7
LETTERS	6	LETTER_LINES	4	LOCATIONS	8544
LOCATION_PRODUCT_AVER AGES	7602	LOCATION_PRODUCT_LEVE LS	8662	MENU_APPS	9
MENU_LINES	332	NAMED_USERS	17	NAMES	2805
NAMES_CATEGORIES	2	NAME_FOR_SUB_LEDGERS	16400	NOTEPADS	90
PARAMETER_HEAD	32	PARAMETER_ITEM	176	PAYMENT_BATCHES	6089
PAYMENT_CATEGORIES	3	PAYMENT_CONTROLS	983	PAYMENT_METHODS	5
PAYMENT_REGISTER	102794	PAYMENT_TERMS	2	PERIODS	7
PERIOD_STOCKS	215886	PL_HEADER_FOOTER	1964849	POP_PRODUCT_TEXT	17
POP_SUPPLIER_TEXT	4	POP_TOLERANCES	1180	POSTABLE_VALUES	5
POSTING_RULES	7	POSTING_RULES_CONTROLS	2	PO_CONTRACTS	33
PO_HEADER_FOOTER	263181	PO_TYPES	14	PRODUCTS	30778
PRODUCT_ALLOCATION	632	PRODUCT_CATEGORIES	99	PRODUCT_CLASSES	8
PRODUCT_GROUPS	3	PRODUCT_RANGES	13	PRODUCT_SUPPLIERS	5329
PROVISIONING	323271	PROVISIONING_ITEMS	2176162	PURCHASE_CONTROLS	2
PURCHASE_LINES	1690979	PURCHASE_ORDERS	382950	REPORT_CONTROL	4
REQUISITIONS	635383	REQUISITION_LINES	735253	REQUISITION_LINE_NOTES	36801
ROLE_ACCESS_RIGHTS	8186	ROLE_AUTHORITIES	22500	ROLE_AUTHORITY_TYPES	1
ROLE_MENU_PRIVILEGES	929	ROLE_PROFILE	631	SAVED_ARGUMENT_HEAD	65
SAVED_ARGUMENT_ITEM	1728	SELF_BALANCING_CENTRES	1	SL_DEL_ADDRESSES	630
STATUTORY_DATES	4	STOCK	8784	STOCK_ADJUSTMENTS	142063
STOCK_BINS	932	STOCK_ISSUES	356103	STOCK_ISSUE_ITEMS	1620642
STOCK_TAKE	15139	STOCK_TAKE_AUDIT	7	STOCK_TRANSACTIONS	1922886
STOCK_TRANSFERS	51	SUB_LEDGERS	5	SUB_LEDGER_MANDATES	10
SUB_LEDGER_TEMPS	232962	SUB_LEDGER_TRANS	810056	TAX_ANALYSIS	301351
TRANSACTION_SUB_TYPES	8	TRANSACTION_TYPES	5	UNITS	106
USER_LAYOUT	7	USER_OPTION	61	USER_PROFILE	1150
VAT_RATES	6	VAT_TYPES	3		

In the case of the transaction tables, these numbers are lower than my initial estimates.

The reason for this is that, in the transaction tables:

- All records give rise to at least 5 clones.
- But only records that fall in the first 4 periods of live running give rise to 53 clones.

The initial published estimates assumed the worst case, that all records would give rise to 53 clones, but the reality has been quite different.

- GL_TRANSACTIONS has come out at less than 200 million records, against the initial estimate of more than 300 million records.
- ACCOUNT_BALANCES has come out at 34 million records against 120 million in the estimate.

However, a ratio of 6 to 1 Transactions to Balances seems more reasonable than the estimate, which was distorted by the transactions used to set up the system.

A number of constraints had to be disabled.

The unique constraint on ACCOUNT_BALANCES was one. The scaling process put records into future periods without regard to whether there were any balances there already. I had a script to de-duplicate the table, but it didn't finish in an acceptable time. This did not affect the test in any way, because the index was still present, only it wasn't unique, the enquiry programs don't care about the duplicates, and the updates all target the current period, which doesn't have any duplicates in it

Some Foreign Key constraints also had to be disabled, e.g that connecting SUB_LEDGER_TEMPS and INVOICE_ANALYSIS.

There were two reasons for this..

- The original extract took place over a working day, and did not attempt a consistent copy. Thus, we had child records in tables extracted late in the day whose parents are not present in the extracts created earlier.
- Some tables do not have periods or dates, for example INVOICE_ANALYSIS. The scaling process operated on each table in isolation, so all records in these tables have had the 53 clone treatment, rather than only those falling in the first four periods of live running.

All these extra records would have to have been deleted before the constraints could be re-enabled.

The records that interfered with our scripts were deleted by hand.

Otherwise, a small improvement in performance resulted from not enabling these constraints.

After capture and scrambling of the seed data off the Live Greater Glasgow system, the processes to perform the database scaling were developed and tested at a Reliable office, using kit borrowed from the Manufacturer. We would only have a 12 day window at the Benchmark Centre, and it was essential that the time at the Benchmark Centre be spent running tests, not constructing a test-bed.

2.4 Script Capture

10gAS-Forms-Services-Finance Version 3.2.1B runs on three physical tiers:

- Client PC
- Application Server/Web Server, (4 processors; 4 of them)
- Database Server (4 processors; 4 of them)

In reality, each of the 2,000 clients will have their own IP address, but the network set-up at the Benchmark Centre favoured approaches that were less profligate in their use of IP Addresses. Thus, we set up a small Linux server on the network as a Web Proxy, pointed our PC client at it, and based our scripts on the traffic from the Web Proxy to the Application Servers that resulted from our usage of the application from the Client PC via the proxy.

Script capture was mostly automated.

- An E2 Systems consultant used an 10gAS-Forms-Services-Finance Version 3.2.1B facility, and kept a running commentary using the E2 Systems PATH e2sync facility.
- The user session and commentary were captured on the Web Proxy Server we supplied
- Other PATH tools then created a re-runnable script from the captured trace, using the commentary to break up the network activity into End-User-meaningful pieces.

Fourteen scripts were captured and engineered in total. The scripts were planned off-site, but version issues meant that we had to re-capture some after we moved to the Benchmark Centre.

A captured script reflects the activity of a single user executing the targeted business process once.

The engineering turns the script into a generic form. It can thus be used by hundreds or thousands of distinct virtual test users which the system under-test cannot distinguish from real physical users.

Each virtual user logged on as a distinct user with their own password. This was a cause of frequent headaches; the users of course have different menus and privileges (there are nearly as many distinct roles as users), and even when the users can actually use the same facilities, the short cut commands may be different; for example, STKS080 was INDENT to some users and STK080 to others...

Each time a virtual user performed a business process it utilised unique data, extracted from the database, or synthesised according to the required input model, as necessary.

This ensured realism.

The scripts that we used in the final tests were as follows.

Script Name	Description
AaAC101P_10	Supplier Query
aaACT041TOP	Drill down from Top level cost centre / second level account
aaACT0412	Drill down from 2nd level cost centre / second level account
aaACT0414	Drill down from 4th level cost centre / second level account
aaACT041B	Drill down from Bottom level cost centre / second level account
aaACT091	Posting Invoices batches
aaPOP016	Purchase Order query
aaSTK054	Stock take data entry
aaSTK080	Provision Entry
aaSTK54E	Bespoke Stock take entry
aaact35o	Invoice batch entry
aapop005	Purchase Order entry
aapop009	Goods Receipt Entry
aastk082	Provision query

A description of the detailed processing steps in each script can be found in Appendix A.

The alert reader will notice that the Pick Processing transaction STKS081has disappeared.

We had trouble working out which Provisions we could use with STKS081. We suspect that some of our users/some of the products are set up to auto-pick on receipt, so that the transaction isn't necessary in these cases. Since someone formerly of Greater Glasgow but now of Public Sector, had indicated that the usage of this transaction we postulated was greater than he expected, we substituted other updates for it.

2.5 The Scenarios Tested

The E2 Systems PATH tools provide us with the ability to generate arbitrary workmixes, scenarios as it were. We were aiming to achieve the scenario set out in the Test Specification, and amplified above, but before we could Receive Goods we had to have Purchase Orders to receive against, and before we could Invoice we needed Receipts, so of necessity we had to run scenarios with a superabundance of Purchase Orders, and then of Goods Receipts, before we could run the Invoices.

We also needed to test the scripts.

Thus, early on we found that we could easily run 2,000 users doing Purchase Orders, we discovered that our version of STKS082 had a bug that rendered it unusable on our large database, and we discovered our confusion over STKS081.

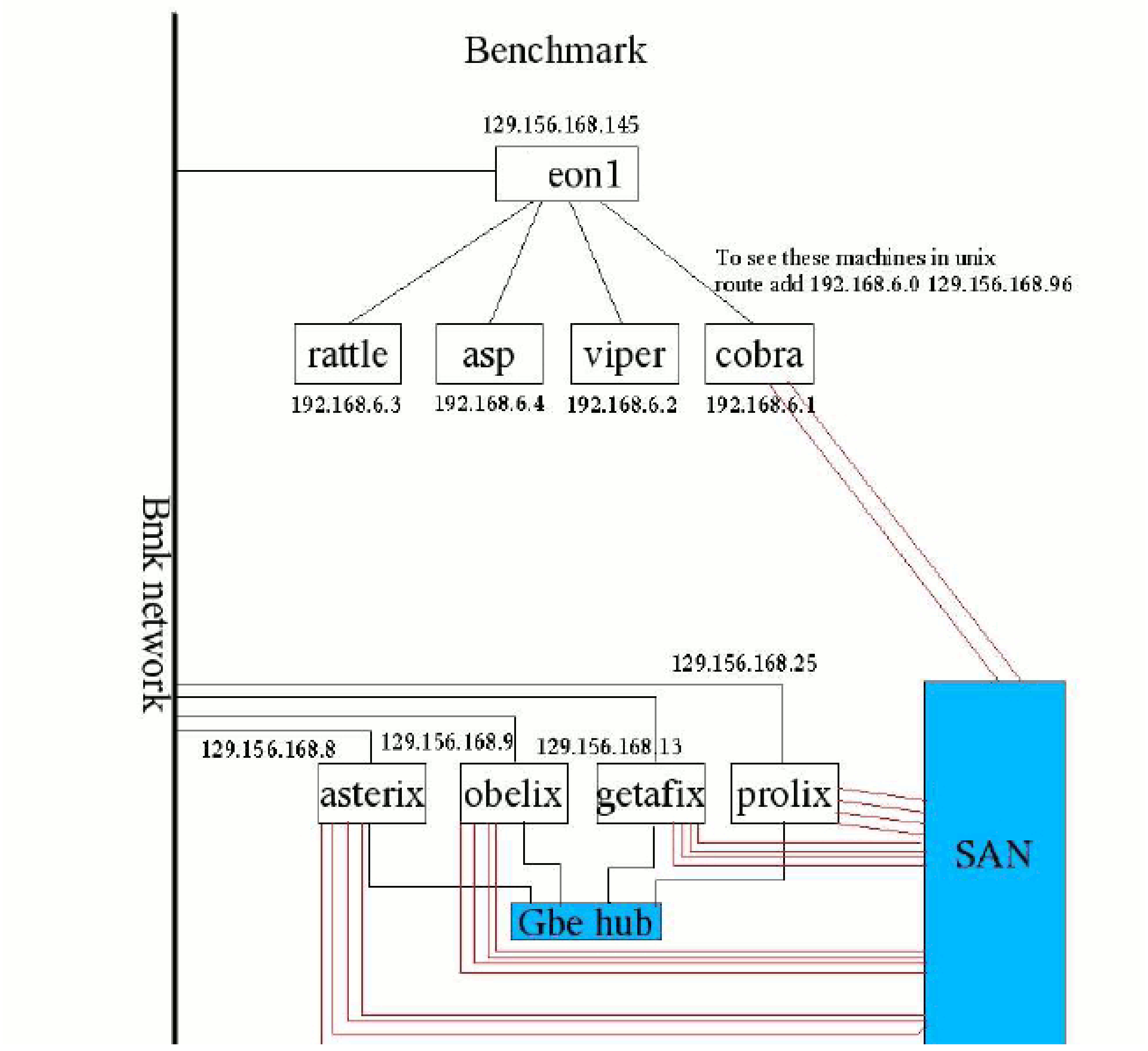
We also found that we couldn't predict exactly how the Purchase Orders would come out of the STKS080 Provisioning, so these orders wouldn't work with our Goods Receipt script.

Thus, we dropped STKS081 and STKS082, and compensated by increasing the number of Purchase Orders entered directly so that there were sufficient to feed the Goods Receipt and Invoice process chain.

However, to fully understand the Scenario that we ran, it is necessary to consider the hardware setup in some detail.

The diagram below shows the hardware set-up that we originally intended to test.

Conceptual Architecture



Towards the bottom of the diagram there are 4 identical servers (asterix, getafix, obelix, prolix) that each run ORACLE database software.

Each of these servers is accessing the same database, on the SAN.

To get good performance from ORACLE, most of the data access has to be serviced from the memory of the database server, DBA's strive to keep this cache hit ratio well above 90 percent. This memory is assuredly not shared.

Thus, these database instances need to co-ordinate with each other. That is the purpose of the Gigabit Ethernet Hub (Gbe hub on the diagram). The multiple database instances exchange lock and cache status information. The amount of information exchanged depends in detail on the application code. Recent modifications to 10gAS-Forms-Services-Finance specifically for ORACLE Real Application Clusters have focused on reducing this traffic.

The multiple database server approach, of course, potentially offers availability benefits, and easy incremental upgrades.

Above the Database Servers are the Application Servers. Note that they are not connected to a Gbe Hub. As configured here, these Application Servers know nothing about each other (although they could be configured differently, as a so-called 'Farm'). They all NFS mount a directory that is used for user report output, and as it happens that was served off one of them, Cobra, but they are to all intents and purposes independent.

The availability and expandability that this approach offers is maximised.

A connection between a PC client and the Application Server has application state. For example, the traffic is scrambled in such a way that if any traffic is injected or dropped, gobbledegook results.

If the Application Servers are configured as a Farm, they update each other with Application State information, but if they are not, they don't. Thus, in this case, once a Client starts a session via an Application Server, they need to stick to that Application Server until the session is over.

The clients do not see the individual Application Servers; they see the Load Balancer. The Load Balancer picks the Application Server for them.

It should be clear from the above that the Load Balancer needs to make sure that once it has chosen an Application Server for a PC Client session, it sticks to it.

Which brings us to the first problem we became aware of.

It was not clear that the Load Balancer was honouring the session 'stickiness' as it was supposed to.

The Manufacturer reconfigured it so that it always put sessions from the same IP Address to the same Application Server.

We could now definitely see that it was load balancing, but we still had failures.

At this point I captured the network packets from a failed session, and was able to find a place in the trace where the Load Balancer had requested that the Client re-transmit a portion of a TCP packet, but forwarded both the original packet, and the re-transmitted fragment, to the Application Server, as independent packets. I know that it did this because the trace has separate Application Server responses for both the original packet and the re-transmitted fragment.

Not surprisingly, the Application Server choked at this point. The error message, of course, depended on what it expected to be happening at the time, and could have been anything. On the basis that a bug in something as fundamental as the load balancing algorithm in a product sold as a Hardware Load Balancer is hard to credit, I think that the earlier apparent 'Load Balancer' failures were also due to this problem. As explained above, the injection of extra packets would produce similar symptoms to a stickiness failure.

The Manufacturer looked suspiciously at our Linux servers, and I looked at network traces in which 40 percent of the packets had been retransmitted; not figures I would expect to see on a healthy LAN.

The Manufacturer was most unwilling to countenance the possibility of problems with the network.

For my part, I now had no confidence in the network connection between our Benchmark work area, and the machine room. I didn't believe the problem could have anything to do with Linux, since the Reliable staff working on the software installation had also experienced failures, and the Linux servers weren't in use at that time.

Our solution was three fold.

The Manufacturer made available to us the six 'Jimi Hendrix Experience' servers to use to drive the test, so they no longer had Linux to upset them.

The 'Jimi Hendrix Servers' were in the machine room, directly connected to the same Gigabit switch that the Application Servers and the Database s ervers hung off, so I no longer had the dubious network segments to upset me.

We decided to address the bug in the Load Balancer by removing it from the test environment altogether. The Network was re-configured so that the Application Servers were addressable independently of the Load Balancer, and we achieved the even distribution of our clients over the network by making our scripts explicitly select their Application Servers.

We are now in a position to explain our final scenario,

E2 Systems Network Benchmark Control Script Sun Nov 6 13:28:17 GMT 2005							
Users	Script	Transactions	Think Time				
12	aapop005	9	12				
10	aapop009	12	16				
20	aaact35o	6	26				
1	aaACT041TOP	3	50				
4	aaPOP016	8	41				
5	aaACT091	15	15				
1	aaAC101P_10	10	5				
3	aaSTK054	11	22				
2	aaSTK54E	13	18				
10	aaSTK080	11	5				
1	aaACT0412	9	17				
3	aaACT0414	12	13				
13	aaACT041B	9	17				
12	viper_aapop005	9	12				
10	viper_aapop009	12	16				
20	viper_aaact35o	6	26				
4	viper_aaPOP016	8	41				
5	viper_aaACT091	15	15				
1	viper_aaAC101P_10	10	5				
3	viper_aaSTK054	11	22				
2	viper_aaSTK54E	13	18				
10	viper_aaSTK080	11	5				
1	viper_aaACT0412	9	17				
3	viper_aaACT0414	12	13				
13	viper_aaACT041B	9	17				
12	asp_aapop005	9	12				
10	asp_aapop009	12	16				
20	asp_aaact35o	6	26				
4	asp_aaPOP016	8	41				
5	asp_aaACT091	15	15				
1	asp_aaAC101P_10	10	5				

E2 Systems Network Benchmark Control Script Sun Nov 6 13:28:17 GMT 2005							
3	asp_aaSTK054	11	22				
2	asp_aaSTK54E	13	18				
10	asp_aaSTK080	11	5				
1	asp_aaACT0412	9	17				
3	asp_aaACT0414	12	13				
13	asp_aaACT041B	9	17				
12	rattle_aapop005	9	12				
10	rattle_aapop009	12	16				
20	rattle_aaact35o	6	26				
4	rattle_aaPOP016	8	41				
5	rattle_aaACT091	15	15				
1	rattle_aaAC101P_10	10	5				
3	rattle_aaSTK054	11	22				
2	rattle_aaSTK54E	13	18				
10	rattle_aaSTK080	11	5				
1	rattle_aaACT0412	9	17				
3	rattle_aaACT0414	12	13				
13	rattle_aaACT041B	9	17				

The table above is a runout file, the file that controls the execution of a tranche of users on a single 'Jimi Hendrix Experience' driver machine, such as 'voodoo', with the columns in the file that are not used in this test, suppressed.

It represents one sixth of the users active during the scenario. Identical files controlled the other 5 members of the 'Jimi Hendrix Experience'.

Looking at line 3 of the table, there were 12 users executing 9 iterations of script aapop005, with a 12 second think time at timing points. This base script targeted Application Server cobra.

Looking further down, one finds asp_aapop005, viper_aapop005 and asp_aapop005. These scripts target asp, viper and rattle respectively.

2.6. Build and Run Benchmark

E2 Systems PATH fully automated this process. First the system performance monitors (netstat, vmstat, iostat, mpstat, sar, process accounting, and the ORACLE AWRRPT facility) were initialised on the Application Servers and the Database servers and then the benchmark was run. The length of the run for each test was between one and two hours, but we can report on any sub-set of the time.

The timings presented later in this report are for a 30 minute period 20 minutes after the start of the test. All the users would have been active at this point.

2.7 Data Analysis and Report Generation

The test results in terms of response times and resource utilisation figures were loaded into a performance database on a PC. This report was then produced.

3. Findings

3.1 Soak Test Success

Apache and Orion (OC4J) on the Application Server, and the ORACLE Database and cluster software on the Database Servers, are the only things that are running all the time, and which may therefore be at risk from memory leaks and so forth.

Except when we had to change the Application Servers' IP addresses, the application servers ran for the entire exercise without ever being shut down, except for one which we restarted whilst we diagnosed what turned out to be an NFS Server failure (see below).

At no stage did we restart the ORACLE database, at any time over the exercise (but see below).

We put the equivalent of a number of more than two full days of work through the systems.

At no stage did we experience problems that were obviously due to memory leaks, or to the elapsed time or the amount of work that had been done in total.

At no stage did we experience runaway (CPU-bound, uselessly looping orphaned processes) on either the Application Servers or Database Servers.

3.2 Sundry Failures

In order to conduct these tests, a great deal of expensive and complicated hardware and software had to be installed and configured in a very short space of time.

The testing threw up a number of failures in this substrate to the application,

All of these problems disrupted the service to clients, disrupted our testing, and indeed disrupted our schedule.

Given the scale of the hardware we were working with, and the fact that we were working with the latest patchsets for all the software deployed, some failures are not perhaps surprising, and of course the likelihood of failures is part of the justification for testing activity.

Public Sector should be interested in these failures. If they occurred whilst building up to live running, they would need to be resolved properly. However, our restricted time window did not allow us to resolve them in a production manner.

3.2.1 ORACLE Database Server Instance Crash

One of the ORACLE Parallel Server instances crashed, once. Nothing was running at the time other than internal ORACLE Database and Enterprise Manager tasks. The trace files suggest that the ORACLE database software somehow lost contact with the ORACLE cluster management software

The impact of this failure on our tests was marginal. To begin with, because of the resilient nature of our configuration, we didn't notice it at all! We ignored the problem, and it didn't recur.

The Manufacturer suggested that they had seen this before with ORACLE ASM. However, I formed the impression that what was really being said was "Of course, if you used OUR Cluster Software, things like this don't happen".

Reliable investigated, and came up with what seems to be a more plausible explanation

Someone signed on to the server as the super-user had issued the command

```
hostname –a
```

when they meant to issue the command

```
uname -a
```

Instead of obtaining the operating system details, they actually changed the name of the host from 'asterix' to '-a'.

If any of the ORACLE components addressed each other using the name of the host, obtained from the host, these commands would fail.

Reliable set the hostname back to asterix before restarting the instance, and we had no further problems, so we think it very likely that Reliable's analysis is correct.

3.2.2 ORACLE Client Core Dump Failures

Occasionally ORACLE clients crashed, giving core dumps; perhaps one failure per hour of testing.

SUN said that they had seen this before when ORACLE Automatic Memory Management was configured.

The crash was probably due to a race condition between the Automatic Memory Management software and the software that manage the Shared Pool; it usually happened when a large PL/SQL object was being loaded.

The impact on our tests was marginal; a small number of user script failures.

Reliable investigated this problem on ORACLE Metalink, the ORACLE support web site, and turned up a suggestion for an ORACLE parameter change, which was applied, and indications that ORACLE were searching for an underlying bug. We can assume that at some point a patch will become available for this.

3.2.3 ORACLE Client Parallel Query Failures

Before going on site at the Benchmark Centre, we found that when the ORACLE Parallel Query option was enabled, ORACLE Clients crashed generating huge trace files (~ 8GByte in one instance) whenever the optimiser tried to use parallel processing to run a query.

The impact was that we couldn't use parallel index building to speed up the loading of the scaled up database, so the database load was slower than it otherwise would have been. Parallel query, offering as it does the opportunity for a single user executing a complex query to completely monopolise a multi-processor machine, to the detriment of all the other users of the system, is a feature that should normally be disabled in an OLTP environment anyway, so we didn't investigate further.

3.2.4 NFS Server Failure

The NFS Server software that shared the users' areas under the report directory crashed, or at least, silently stopped working, once. This turned out to be a single point of failure; the system became inaccessible until the NFS service was stopped and restarted.

The impact of this was that we completely lost the test run that we had set up overnight to run multiple streams of reports on each server against the interactive workload, and a further couple of hours whilst we tracked down what was wrong.

Our workaround, once we had realised what the problem was to restart the NFS Server.

Reliable response to this single point of failure issue is to be found in section 4.2.

3.2.5 Load Balancer TCP/IP Bug

The Load Balancer TCP/IP stack turned out to have a serious bug. At least some of the time that a TCP retransmission was requested of a client, both the original packet and the retransmitted packet were forwarded to the ORACLE Application Servers, but as sequential packets rather than as the same packet sent twice. As a result of this problem client sessions could fail at any time, and emit diagnostic output that made no sense, except in the light of what turned out to be the reason.

We cannot say if all the failures we experienced when using the Load Balancer were merely symptomatic of this problem, or represented different problems altogether.

In terms of time lost during the hardware availability window at the Benchmark Centre, this was the most serious. All activity from the moment that the hardware commissioned until we had the network trace that demonstrated that the Load Balancer didn't work properly, 8 days later (almost 2/3 of all the time available in the Benchmark Centre) was disrupted. The problem affected both Reliable work on installing and commissioning the software, and E2 Systems efforts to develop test scripts.

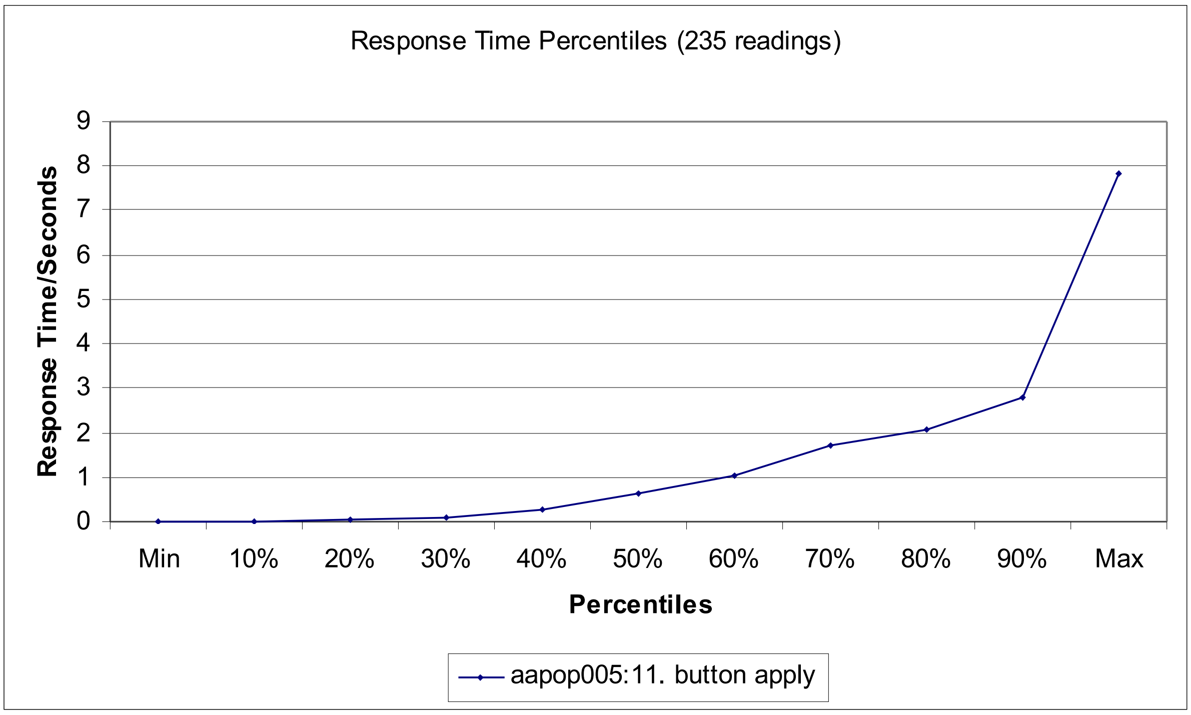
It did not help that the Benchmark Test Centre Load Balancer expert was, due to difficult personal circumstances, unavailable throughout. In the light of what turned out to be the nature of the problem, he was unlikely to have been able to provide a fix for it, but his input may have helped us to come to the decision to abandon the Load Balancer sooner.

3.3 User Response Times

Even though we discovered opportunities for improvement, Reliable 10gAS-Forms-Services-Finance, delivered more or less as it is, on this hardware platform, would have no difficulty with the OLTP aspects of the projected Scottish Public Sector Shared Services workload. Whilst the test transaction throughputs recorded did not reach our target 'Busy Hour', they exceed the rates needed for the Public Sector to get through its workload in a day by 50 percent, so I am comfortable with the assertion that the hardware tested would be an adequate platform for the Public Sector. If the improvements outlined in section 4.2 were made, I would expect that our Busy Hour throughput goals would be achievable.

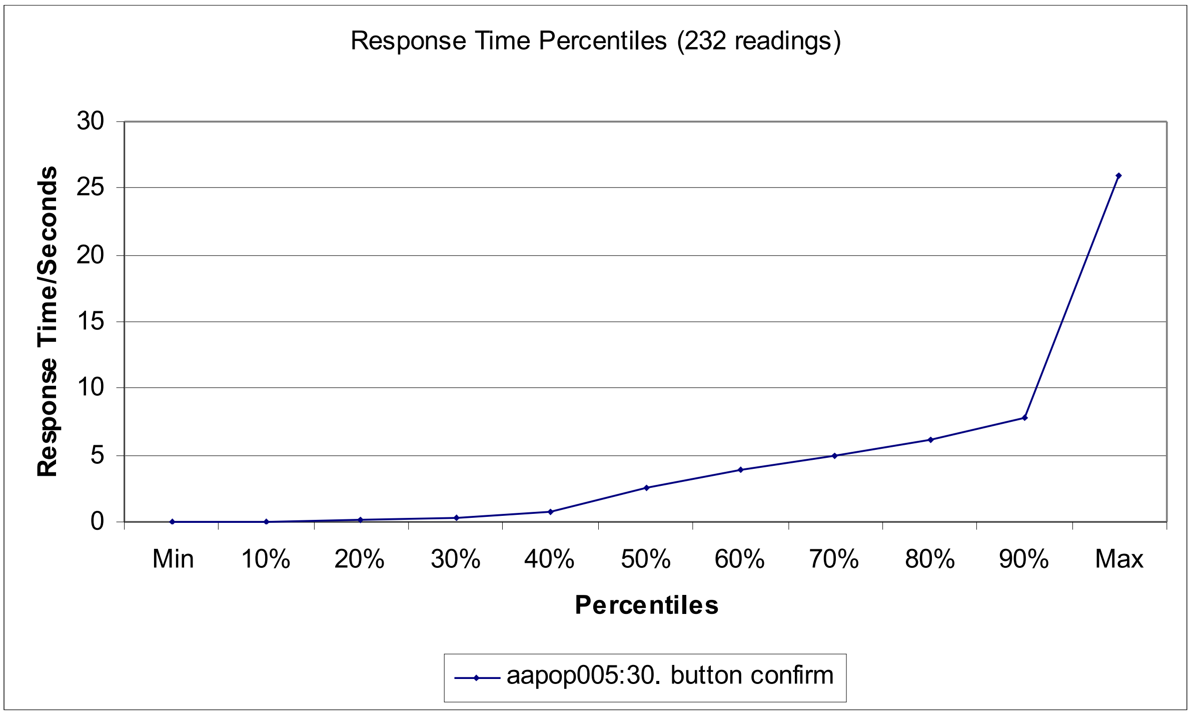
The response times meet our criteria for technical success.

The graph below shows the response time percentiles for creating a Purchase Order Header, an example of a simple update.



The success criterion is that the 70th percentile should be less than 2 seconds, and it is.

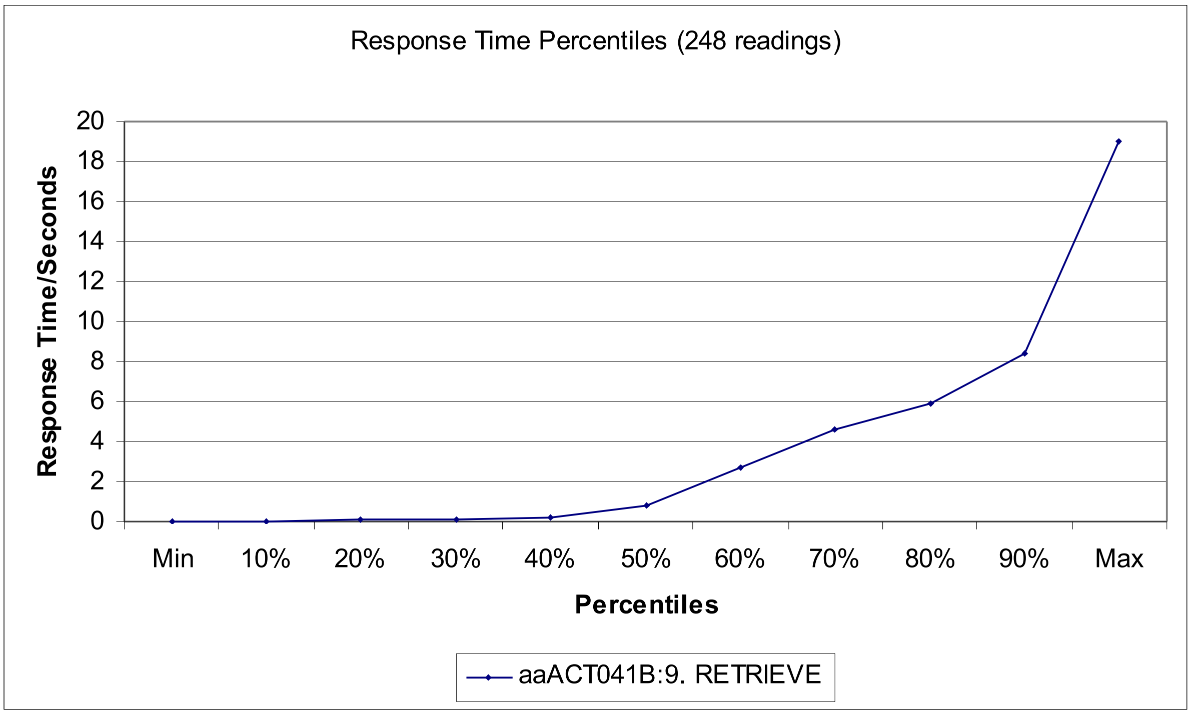
The following graph shows the response time for Purchase Order Confirm of a 4 line Purchase Order, through Application Server cobra.



This is an example of a complex update. The success criterion is that the 70th percentile response should be less than 5 seconds, and it is.

These are the readings for a single Application Server in half an hour, so we are looking at an hourly rate of 4 * 232 * 2 = 1,826.

The graph below shows the response times for a complex query, the ACTS041 Retrieve for a Bottom Level (Postable) Cost Centre and a Second Level Account.



The bottom level cost centre/second level account ACT041 retrieve typically pulls back 100 records. The rate here is 1872 per hour.

The success criterion is that the 70th percentile response should be less than 10 seconds, and it is.

Overall, we achieved around half the target throughput, which was three times the long term average, so we exceed the capacity needed to get through a day's work in a day by around 50 percent.

The scenario hot spot is the ACCOUNT_BALANCES table.

Goods Receipts and Invoice Posting update Account Balances.

The year-to-date ACTS041 enquiries potentially target the same records.

The Sub-Ledger Control Accounts are particular choke points.

ORACLE shows users records as they were before uncommitted updates were applied to them.

Thus, ORACLE is having to ship ACCOUNT_BALANCE locks and 'Consistent Read' restored ACCOUNT_BALANCE blocks all around the cluster.

In all the tests that we ran with large numbers of users with a mix of update transactions, enquiries and reports, the database servers ran at 100 percent busy, with run queues of 20 or more (see below).

The response times for our simple, medium and complex queries were nevertheless acceptable, as above.

However query times for what we might collectively term ‘super-complex’ enquiries became greatly extended.

1 1/2 hours for a top level ACTS041 was the longest we recorded that actually finished within one of our test windows.

This single enquiry summarises perhaps 1,500,000 balances, as opposed to the 100 or so balances picked out by the Bottom Level Cost Centre/Second Level Account queries.

We needed to change some indexes in order to get the ACTS041 queries as fast as they were. We added an index to target the Cost Centre in ROLE_ACCESS_RIGHTS (ROA_PARAMETER_2, ROA_PARAMETER_1, ROA_TYPE) and an index on ACCOUNT_BALANCES(AB_COST_CENTRE,AB_PERIOD_ID, AB_ACCOUNT), and these made a big difference (40 minute response times reduced to a few seconds for some ACTS041 queries, though not all. Many ACTS041 queries still resulted in full table scans.). We spotted some opportunities for refinements to the code as well, which are highlighted in section 4.2 below. However, there is simply no way that a query that summarises 1,500,000 records on an OLTP system can be expected to return results in a handful of seconds.

To get response times of a few seconds, a query has to target far fewer records. Which these might if some of the potential results were somehow pre-calculated. ORACLE provides a method by which this may be achieved, the Materialised View. They are usually used in Management Information databases, but it is not unheard of to make use of them in OLTP schemas. I added this as a suggestion to the list in section 4.2.

Easier to deal with, but no less hungry in terms of the system resource requirements, was the dearth of indexes on the cost centre columns that identify warehouses or simply restrict the viewing rights of the user.

Generalising somewhat, it appeared that any report that was specified to be run once for a cost centre or warehouse would execute a full table scan against one of the large tables.

The way that Greater Glasgow work, whenever a Purchase Order is processed; it is printed.

This was O.K.

A pop_r05 Purchase Order Print report took a small fraction of 1second to run, regardless of the number of users.

Whenever Goods were Received, a Goods Receipt note should be printed.

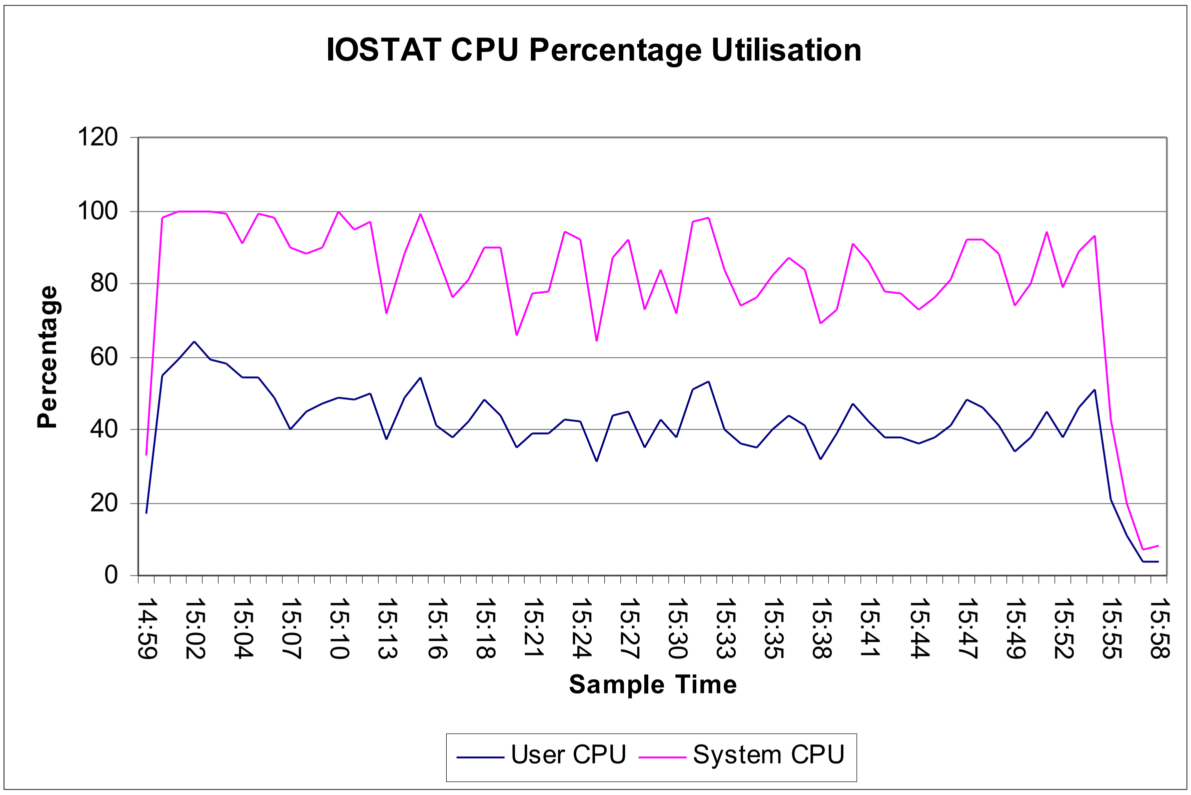
This was not O.K.

A full table scan of millions of rows resulted.

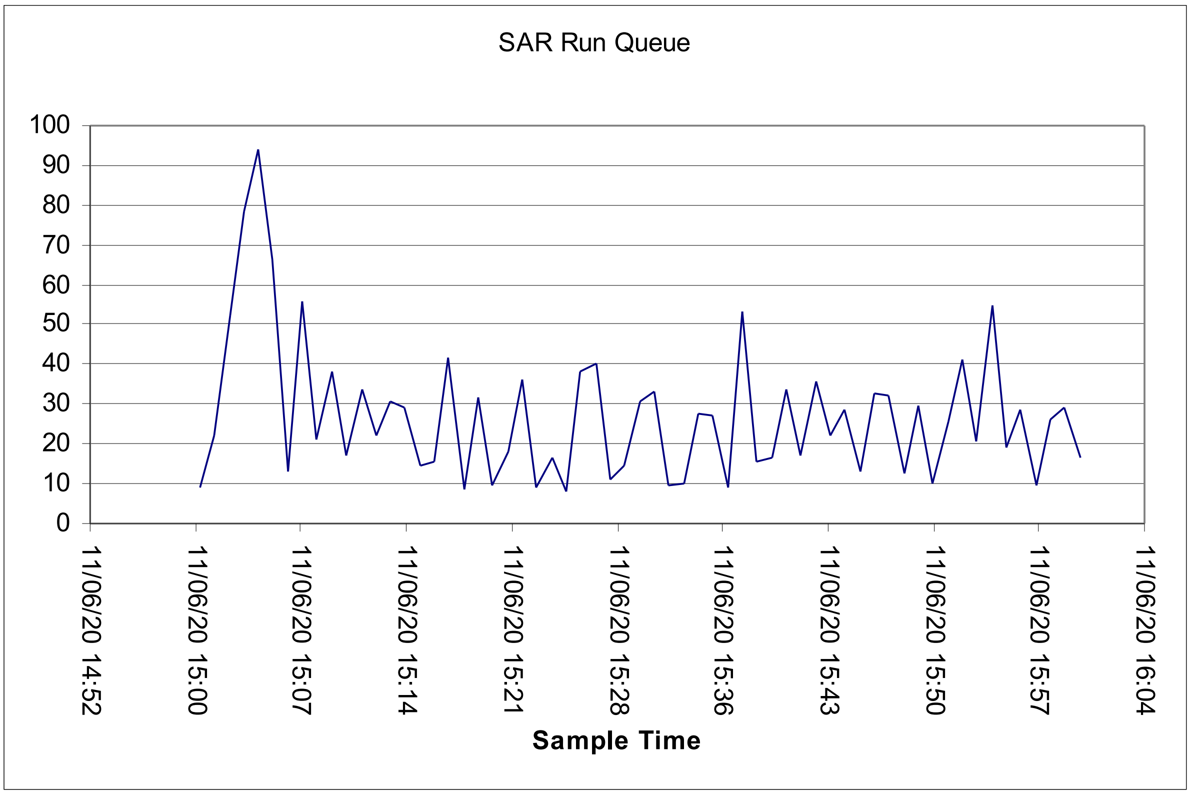
Similar results held, for example, for REQ_R22, and a number.

3.4 *Application Server Resource Consumption Profile*

The Application servers are busy, but are apparently not quite max'ed out, as the graph below for one of the Application Servers, asp, illustrates (the graphs for all the application servers look very similar).



However, when the run queue is examined, it looks as though it is max’ed out.



The Application Server asp has only 4 CPU's, so if the run queue is permanently in double figures, one might expect the CPU to be 100 percent utilised.

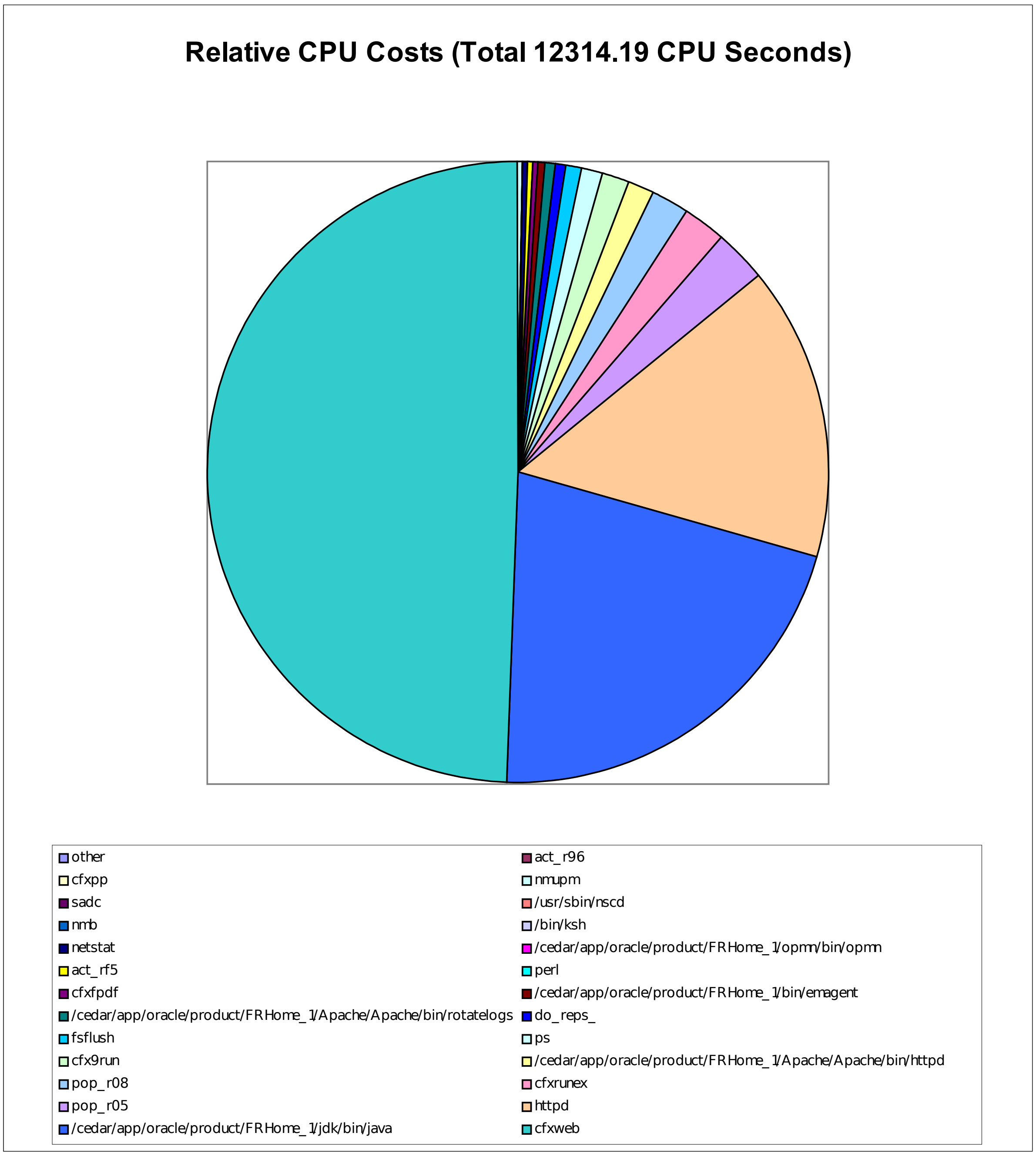
If the run queue greatly exceeds the number of CPU's, but CPU utilisation is not 100 percent, it suggests that there are processes waiting for spin locks, spinning and sleeping.

The graph below shows the processes that consume CPU resource on the application server.

The lion's share is taken by the web forms process, cfxweb.

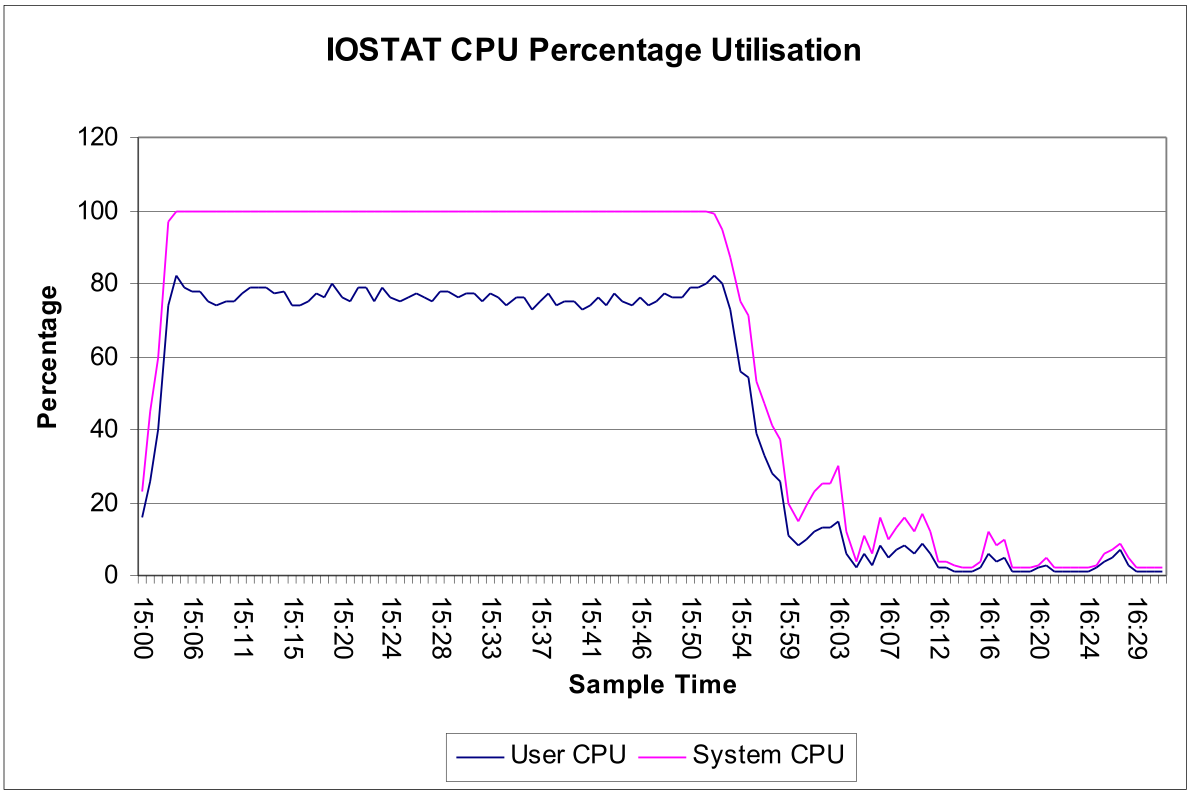
Most of the rest is taken by java, and apache (httpd).

My guess is that the spinning and sleeping is occurring within java. We were running with a single Forms servlet on each application server. I speculate that we could reduce the spinning and sleeping behaviour, and incidentally reduce the length of time to log in, by increasing the number of forms servlets and forms listeners, and having some pre-spawned.

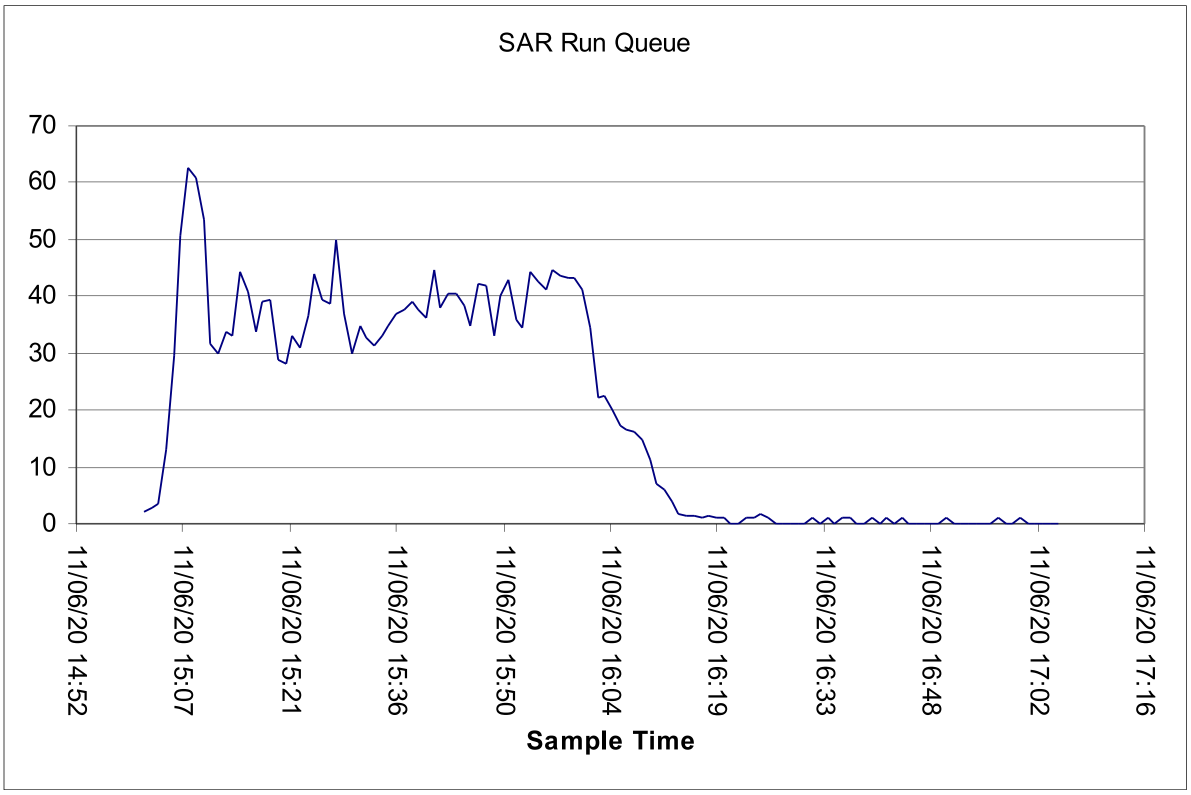


3.5 *Database Server Resource Consumption Profile*

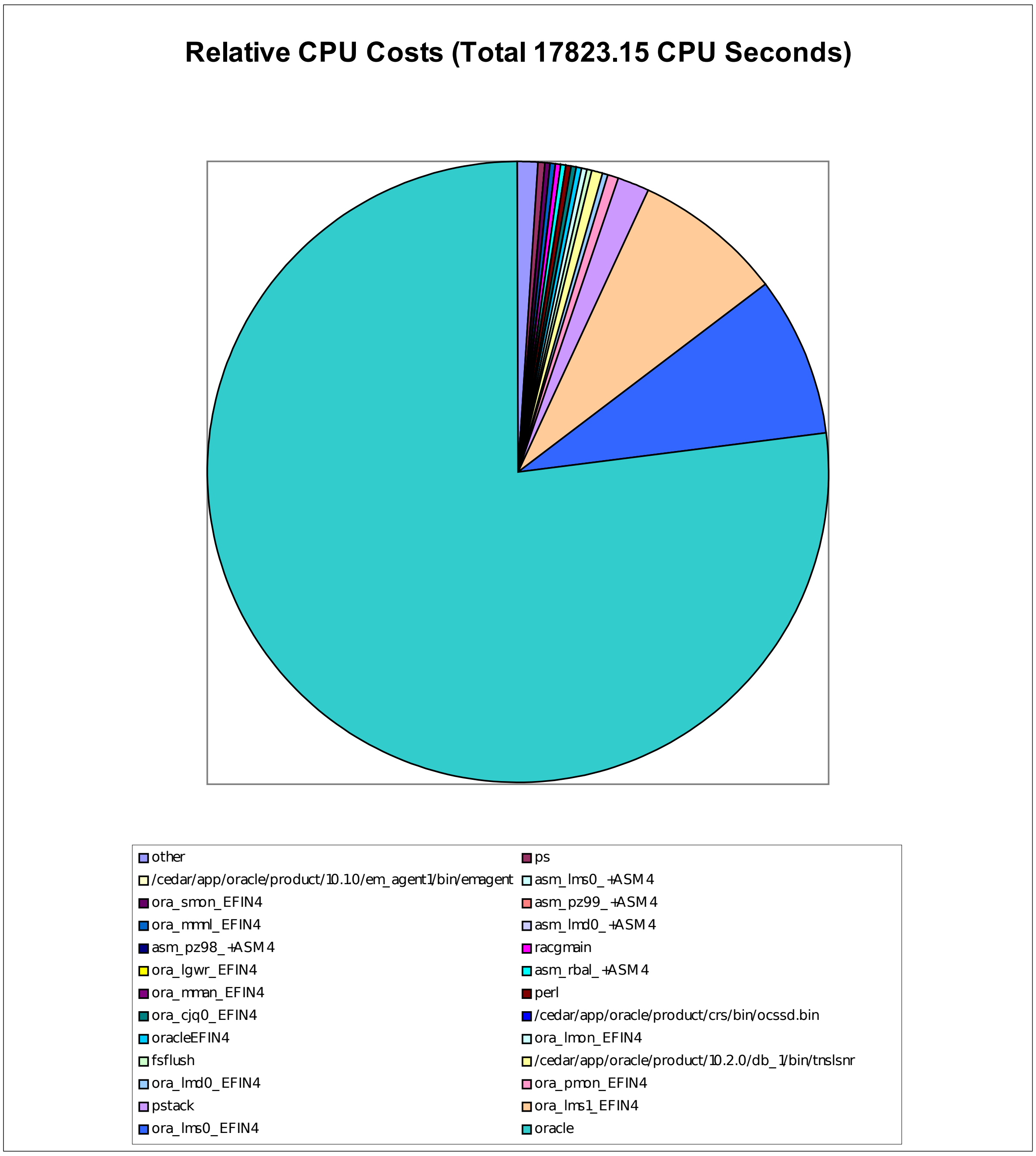
The database server is maxed out throughout the test run, as can be seen in the CPU utilisation graph below from asterix, one of the database servers (the graphs for all the database servers look very similar).



In contrast to the Application Servers, the CPU utilisation and the Run Queue are entirely in agreement. Over the time period that the reported response times relate to, there were between 8 and 10 runnable processes available for each of the 4 CPU's, and those CPU's were 100 percent occupied.



The Process Accounting information shows that much the greatest part of the CPU resource is consumed by ORACLE processes access the database on behalf of virtual users. However, the two next largest pieces, amounting to around 20 percent of the time consumed by the ORACLE user agent processes, was consumed by the processes that co-ordinate with the other database instances in the cluster.



3.6 *Break Test Success*

It can be seen above, in sections 3.4 and 3.5, that in our attempts to put the planned busy hour workload through the systems, we drove both the Application Servers and Database Servers to 100 percent CPU utilisation. So what does the break test mean, in this context? We cannot have more than 100 percent utilisation. In one test we had 5 streams of reports running continuously from each Application Server, in addition to the 2,000 users, and in another we had 2,200 users connected (the maximum number that the installed 10gAS-Forms-Services-Finance licence allowed). The CPU utilisation stayed the same, the response times for simple tasks remained acceptable, and the response times for the ultra-complex tasks, things that would take minutes on an otherwise empty set of machines, became greatly extended. We deduce from this that the behaviour of the system remains stable under extreme stress and performance degrades gracefully, i.e. users can continue to use 10gAS-Forms-Services-Finance, however response times increase.

4. Conclusions

4.1 Overall Assessment

The test scenario called for Purchase Orders, Goods Receipts and Purchase Invoices to be processed. In order to build up a pipeline of inputs for the later stages, we ran early tests with 2,000 users just entering Purchase Orders (POPS005), and then tests with Purchase Orders (POPS005) and Goods Receipt notes (POPS009). We had no difficulty getting 2,000 users to process 10 documents each, as specified; Application Servers would be 30 percent busy, and the database servers perhaps 50 percent busy tops. Response times were excellent.

Comparison of the CPU resource consumed by ORACLE agent processes, and the processes dedicated to inter-instance communication, suggest that the Parallel Server overhead (as compared with running a single ORACLE instance) was between 5 and 20 percent, depending in detail on the exact mix of transactions.

However, as we added a wider variety of transactions and reports, we encountered programs that were coded in such a way that they would not use Indexes to evaluate the query, even though (for correctly coded programs) indexes were available, and should have been used. Examples were POPS016, STKS082, POP_R09, ACTS035.

Some of the Level 1 and Level 2 Cost Centre/Level 2 Account enquiries in the workmix summarise so many records that reading all the 35 million records in ACCOUNT_BALANCES may well be the best way of evaluating these queries. And we have up to 200 million rows in the transaction tables. Overall, the system resources consumed by programs that did not use indexes to identify the handful of rows to be displayed or printed made the scenario throughput goals (3 times the long term average) unattainable.

In all the tests that we ran with large numbers of users with a mix of update transactions, enquiries and reports, the database servers ran at 100 percent busy, with run queues of 20 or more.

Nevertheless, update response times remained workable, even though query times for what we might collectively term ‘super-complex’ enquiries became greatly extended.

In conclusion, the testing has demonstrated.

- Some failures in the hardware, operating system, database management system substrate; not many considering the short time over which the systems were put together.
- The failures notwithstanding, Reliable 10gAS-Forms-Services-Finance runs perfectly well on ORACLE Real Application Clusters.

To re-iterate, even though we discovered opportunities for improvement, Reliable 10gAS-Forms-Services-Finance, delivered more or less as it is, on this hardware platform, would have no difficulty with the OLTP aspects of the projected Scottish Public Sector Shared Services workload. Whilst the test transaction throughputs recorded did not reach our target 'Busy Hour', they exceed the rates needed for the Public Sector to get through its workload in a day by 50 percent, so I am comfortable with the assertion that the hardware tested would be an adequate platform for the Public Sector. If the improvements outlined in section 4.2 were made, I would expect that our Busy Hour throughput goals would be achievable.

Since

- A system comparable to that proposed for Public Sector was put together and commissioned
- The system throughputs exceeded the levels required for the system to get through its on-line work in the on-line day.
- The End User Response Times satisfied the Technical Success Criteria.
- The Break test demonstrated that 10gAS-Forms-Services-Finance performance degrades gracefully as the number of concurrent users increases
- The Soak test demonstrated that 10gAS-Forms-Services-Finance remains stable under continuous load

we have no hesitation in asserting that the Performance Test establishes the viability of the proposed, cost-effective, horizontally scalable hardware solution.

4.2 10gAS-Forms-Services-Finance Performance Improvement Opportunities

During the testing, we identified a number of opportunities to improve the performance of 10gAS-Forms-Services-Finance. These are listed in the following table, together with the Reliable response. Reliable high-level response to the issues raised below that relate to 10gAS-Forms-Services-Finance is as follows:

The issues detailed below will be logged as part of this project and given a Priority 1 classification internally. This means that they must be addressed or resolved as part of the 10gAS-Forms-Services-Finance 3.3.S1 product release. In addition, planned 10gAS-Forms-Services-Finance 3.3.S1 testing will cover these identified areas. Finally, Reliable will review the points made below in conjunction with the functional specification to ensure that any optimisations will not affect the business processes in 10gAS-Forms-Services-Finance except to improve performance.

Opportunity	Reliable Response
There is Form Block co-ordination code in POPS016, STKS082, ACTS035 that disables indexes.	Will be logged internally as an issue to be resolved for 10gAS-Forms-Services-Finance 3.3.S1 core product.
There is code used in reports that handles multiple user input options with code that disables indexes on all of them, in particular POP_R09.	Will be logged internally as an issue to be resolved for 10gAS-Forms-Services-Finance 3.3.S1 core product.
An index on ROLE_ACCESS_RIGHTS (ROA_PARAMETER_2, ROA_PARAMETER_1, ROA_TYPE) significantly improves performance for non super users.	This index will become part of 10gAS-Forms-Services-Finance 3.3.S1 core product.
The failure of the ACTS041ACCOUNT_BALANCES query code to allow ORACLE to use the indexes that naturally join ACCOUNT_BALANCES with the account segment hierarchy tables is a hangover from earlier versions of ORACLE.	Will be logged internally as an issue to be resolved for 10gAS-Forms-Services-Finance 3.3.S1 core product. Reliable continually strives to take advantage of the latest Oracle optimisations and will do so in this case as well.
Likewise the code that implements AB_PERIOD_ID between x and y as AB_PERIOD_ID= a or AB_PERIOD_ID = b … or AB_PERIOD_ID = z is also a hangover from old versions of ORACLE.	Will be logged internally as an issue to be resolved for 10gAS-Forms-Services-Finance 3.3.S1 core product. Reliable continually strives to take advantage of the latest Oracle optimisations and will do so in this case as well.
ACTS041 queries from high up in the hierarchy may summarise millions of records. There may be a possibility that by using materialised views, these numbers could be reduced, at the expense of the resources needed to maintain the materialised views.	Will be logged internally as an issue to be resolved for 10gAS-Forms-Services-Finance 3.3.S1 core product. Reliable continually strives to take advantage of the latest Oracle optimisations and will do so in this case as well
Public Sector will have hundreds of warehouses. Many related reports work a warehouse at a time, Currently, no indexes exist to support working a warehouse at a time.	Reliable will examine the built-in reports and where indexes will improve warehouse access performance, they will be added.
An index on ACCOUNT_BALANCES(AB_COST_CENTRE,AB_PERIOD_ID, AB_ACCOUNT) led to significant	This index will become part of 10gAS-Forms-Services-Finance 3.3.S1 core product.

reductions in query elapsed times for ACTS041	
Reliable 10gAS-Forms-Services-Finance would give better performance on a large databases if all of the database accesses that are part of the normal Business Processes, and which precisely identify small numbers of records in tables of the size that we anticipate, use indexed access paths.	Reliable have dedicated a project phase in 10gAS-Forms-Services-Finance 3.3.S1 to examine the performance of database access paths, with the feedback from this report a primary driver in identifying the highest profile candidates for optimisation.
The NFS Server that provides access to the user report output constitutes a single point of failure,	There are multiple approaches available to remove this single point of failure of the system. NFS itself can be configured to fail-over gracefully to another node. In addition, Reliable recommends that Partner explore the possibility of locating the user directories on a SAN or similar which will provide the required levels of fail-over and fault tolerance.
Purchase Order Number (and related values, such as the Requisition Number) need to be increased in width, as do the Stock Transaction, GL and sub-ledger reference columns, to cope with the anticipated volumes of transactions.	Reliable is aware of the need to increase the width of certain transaction numbers in the system and has already scheduled this work as part of 10gAS-Forms-Services-Finance 3.3.S1.
If you use POPS016 on a Purchase Order with a manual order number (as we had to use to allow us to fit all the required Purchase Order Numbers in the current column width) the form executes an open query against the Purchase Order Lines when you go to view the Purchase Order Lines.	This screen will be performance reviewed to examine the purpose of the open query and to ascertain if it can be optimised.

Appendix A – Scripts

Every script begins with a logon process:

1. START aaAC101P_10 \
2. logon pmyz \
.
.
.
74. close appl \

aaAC101P_10 : Supplier Query

3. fastpath AC101P \
4. suppl=YXX3337 \
5. Qry supplier \
6. button ALL POSTED \
7. their ref = 817089 \
8. qry posted items \
9. close form \
10. enter qry mode \
11. supplier=YXX3337 \

aaACT0412 : Drill down from 2nd level cost centre / second level account

3. fastpath ACT041 \
4. CC2=E04345 \
5. ACCT2=CXCZQW \
6. start=1 \
7. end=12 \
8. col2=budget1 \
9. RETRIEVE \
10. EXPLODE1 \
11. EXPLODE2 \
12. EXPLODE3 \
13. EXPLODE4 \
14. EXPLODE5 \
15. EXPLODE6 \
16. EXPLODE7 \
17. EXPLODE8 \
18. QUERY movements \
19. EXPLODE \
20. CLOSE \
21. CLOSE \
22. CLOSE \
23. CLOSE \

aaACT0414, aaACT041B and aaACT041TOP

These three scripts are similar to the Drill down enquiry above. However the utilize cost centres at the top, fourth and bottom levels. The accounts are second level in al;l casas.

aaACT091 : Posting Invoices batches

4. fastpath ACT091 \
5. sl=FW6 \

6. button DETAILS \
7. ackn \
8. button ALL \
9. ackn \
10. batch=B006215 \
11. QRY \
12. tick to post \
13. vbutton POST \
14. ackn q for post \
15. close form \

aaPOP016 : Purchase Order query

3. fastpath POP016 \
4. button POs \
5. supplier=YXX3337 \
6. orderdate=07/06/04 \
7. Query PO \
8. Qry PO Items \
9. close form \
10. close form \

aaSTK054 : Stock take data entry

3. fastpath STK054 \
4. enter qry mode \
5. seq=1238 \
6. qry stock take \
7. tab entry lines \
8. enter qry mode \
9. prod1=0232258 \
10. qry prod1 \
11. qty=1 \
12. button apply \
13. ackn update \
14. close form \

aaSTK080 : Provision Entry

3. fastpath INDENT \
4. ref=ZZA00009 \
5. authoriser=gneh \
6. prod1=0553586 \
7. qty=1 \
8. goto cc \
9. fastfind \
10. cc1=016998 \
11. qry \
12. button OK \
13. newline \
14. prod2=0641624 \
15. qty=1 \
16. goto cc \
.
.
For seven provision lines
.
.

57. fastfind \
58. cc7=016998 \
59. qry \
60. OK \
61. button apply \
62. button confirm \
63. button close \
64. button appl \

aaSTK54E : Bespoke Stock take entry

3. fastpath STK54E \
4. enter qry mode \
5. seq=1237 \
6. qry stock take \
7. tab entry lines \
8. enter qry mode \
9. prod=120415 \
10. qry product \
11. qty=1 \
12. button apply \
13. ackn update \
14. close form \

aaact35o : Invoice batch entry

3. fastpath act35o \
4. batch name nh67678 \
5. sl=fw6 \
6. button apply \
7. tab invoice \
8. ponum=vm09906 \
9. transdate=01/10/05 \
10. button apply \
11. button match all \
12. net=8 \
13. vat=1.4 \
14. button apply \
15. button authorise \
16. tab batch \
17. gross=9.4 \
18. qty=4 \
19. tick to be posted \
20. button apply \
21. close form \

aapop005 : Purchase Order entry

3. fastpath pop005# \
4. order type=vm \
5. ref1=nh12345 \
6. authoriser=gncw \
7. sl=fw6 \
8. supplier=cvp3336 \
9. cc=016998 \
10. ref2=nh12345 \

11. button apply \
12. tab lines \
13. prod1=0699300 \
14. tab thru acct \
15. qty=1 \
16. newline \
17. prod2=1652713 \
18. tab thru acct \
19. qty=1 \
20. newline \
21. prod3=2111333 \
22. tab thru acct \
23. qty=1 \
24. newline \
25. prod4=3211433 \
26. tab thru acct \
27. qty=1 \
28. button apply \
29. tab order \
30. button confirm \
31. close form \
32. close appl \

aapop009 : Goods Receipt Entry

3. fastpath pop009 \
4. ponum=vm09904 \
5. del note=123456 \
6. button apply \
7. tab lines \
8. qty=1 \
9. downline \
10. qty=1 \
11. downline \
12. qty=1 \
13. downline \
14. qty=1 \
15. button apply \
16. tab main \
17. button confirm \
18. close form \

aastk082 : Provision query

3. fastpath=stk082 \
4. ref=38024 \
5. exit qry \
6. down blk \
7. close form \
8. close appl \
9. close ie \

Appendix B – Hardware Configuration

Removed.