A CICS/DB2 Legacy Application Takes the Sysplex Plunge

Jodi Perry
William J. Raymond
David J. Young
Amdahl Corporation

What are the issues involved in converting a single region CICS/DB2 application into the multiple regions needed by CICSPlex/SM? What application code changes are required? What is the cost of adding DB2 datasharing? How does application design influence the cost of DB2 datasharing? This paper attempts to answer these and other questions by describing the steps taken to upgrade a CICS/DB2 legacy application to include CICSPlex/SM and DB2 data sharing, and identifying the associated increases in resource consumption.

INTRODUCTION - INITIAL RESULTS

The target application is used by a public utility to check the credit status of current and potential customers before scheduling service updates. The application is written mainly in COBOL, with some ASSEMBLER routines, and was originally designed for use with VSAM and standard 3270 type character based displays. Over the years, VSAM was replaced by DB2, and the 3270 displays were enhanced with a Windows/GUI front end.

The application consists of online and batch components: a lot of the online work is saved for later processing at night by the batch component, with the typical time constraints on the batch window. If the batch window processing is delayed, or extended, availability of the online system may be delayed.

There are several reasons for investigating CICSPlex/SM and DB2 datasharing for use by this application. One reason is recoverability: If one image fails in the Sysplex, the application should be able to keep running on a different image, assuming all the restart/recovery procedures are successfully implemented. Another reason is the potential reduction in elapsed time provided by DB2 datasharing: If one image is particularly CPU constrained, DB2 work could be directed to a different image with CPU cycles to spare. Or better yet, two or more members of the datasharing group could do DB2 work in parallel. This could in turn lead to reduced batch window time, and thus allow more up-time for the online portion of the credit checking application.

On a more global basis, perhaps the most significant reason for investigating CICSPlex/SM and DB2 datasharing is to check the viability, performance, and scalability of online production applications in a full function parallel Sysplex environment.

As important as improved recoverability and availability are to this application, it is equally important to know the related resource costs of these features. This paper discusses, in detail, the performance characteristics of six different online environments, all operational under MVS/ESA SP5.2.2 running the Workload Manager. The first implementation and base case (SR) was a single CICS region, no CICSPlex or datasharing. Second (T/A) was a TOR/AOR implementation, no CICSPlex or datasharing. Third (T/A CP) was a TOR/AOR implementation with CICSPlex/SM, no datasharing. Fourth (T/A/A/CP) was a TOR/AOR/AOR implementation, using the dynamic transaction routing facility of CICSPlex/SM. Fifth (T/A/A/CP/DS1) was the same implementation as four, with one-way datasharing. Sixth (T/A/A/CP/DS2) was the same implementation as five, with two-way datasharing. Figure 1 graphs the CPU seconds per transaction (commit) for the six environments.

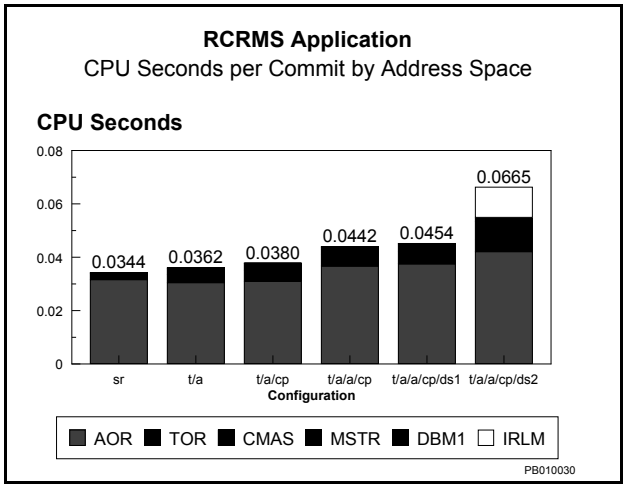

Figure 1 - CPU Seconds per Transaction

For this application, adding CICSPlex processing and dynamic transaction routing using MVS Goal mode processing costs approximately 28% more CPU per transaction than the base case. Adding two way DB2 datasharing costs an additional 65% more CPU per transaction. The final implementation with CICSPlex/SM and two way DB2 datasharing costs almost twice the CPU per transaction as the base case.

The CPU seconds per transaction are calculated by summing the CPU times as reported by the RMF Workload Manager for the various application address spaces (CICS application regions, CICS CICSPlex regions and DB2 system address spaces) and dividing by the number of commits for the measurement interval.

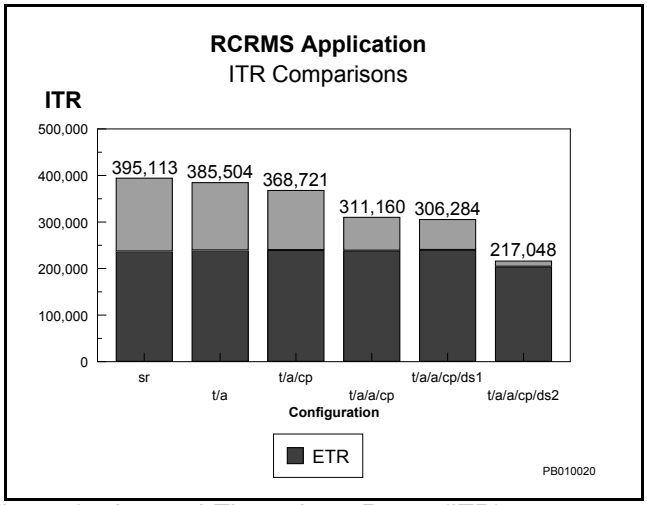

Figure 2 - Internal Throughput Rates (ITR)

For each test, 1,000 users were logged on and a transaction driver (TPNS, Teleprocessing Network Simulator) created transactions which were ramped up to issuing approximately 2,600 transactions per minute (43 transactions a second) over a 75-minute measurement interval. This transaction arrival rate is approximately five times greater than the anticipated load for the system. The rate was increased fivefold in an attempt to plan for unseen contingencies by stressing the system beyond any reasonable expectation. Figure 2 graphs the ITRs for the six environments.

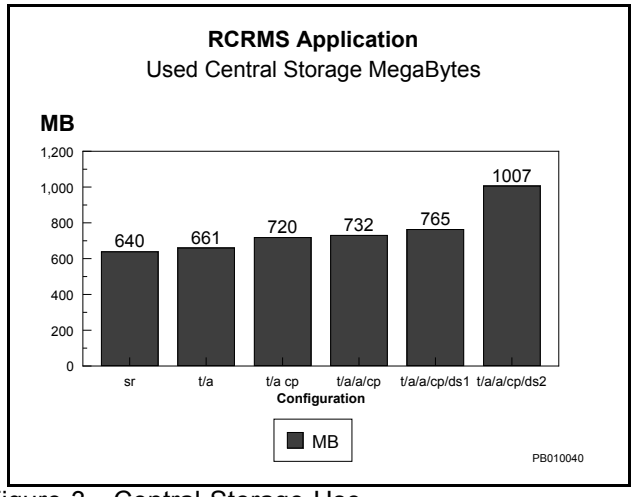

Figure 3 - Central Storage Use

Notice the ITRs in Figure 2 decrease at about the same rate as the CPU per transaction increases in Figure 1. The CICSPlex/SM environment with two-way DB2 datasharing produces approximately 45% fewer transactions (ITR) than the base case. Compare this with the 93% increase in CPU per transaction for the same environments, as shown in Figure 1.

The added functionality provided by CICSPlex/SM and DB2 datasharing also increases real storage use. Figure 3 graphs the average allocated central storage size for all six environments, indicating a 57% increase in central storage for the final implementation versus the base case.

The remainder of this paper discusses each environment in detail, and gives an overview of the application and its operating environment.

OPERATING ENVIRONMENT

Figure 4 shows a high level overview of the hardware configuration of the testing environment. It consists of seven MVS images (six in the Sysplex) housed on an IBM 9021-9X2, an IBM 9672-R64, an IBM 9672-RX3, and an Amdahl GS585. Two standalone IBM 9674-C02s, each with two engines and one gigabyte of memory were used as coupling facilities. Two coupling facilities were used to test recoverability in the event one of the coupling facilities failed.

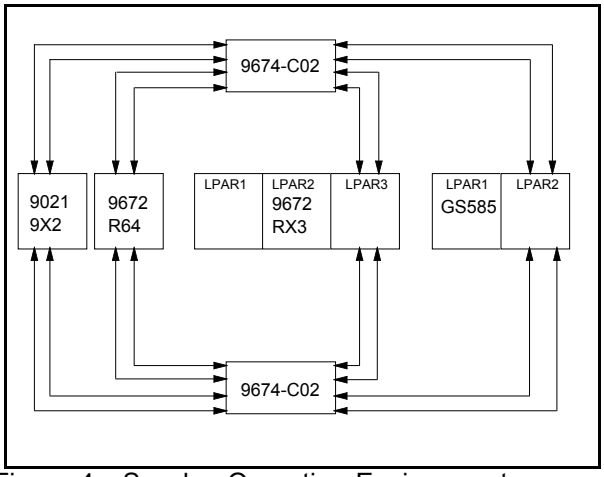

Figure 4 - Sysplex Operating Environment

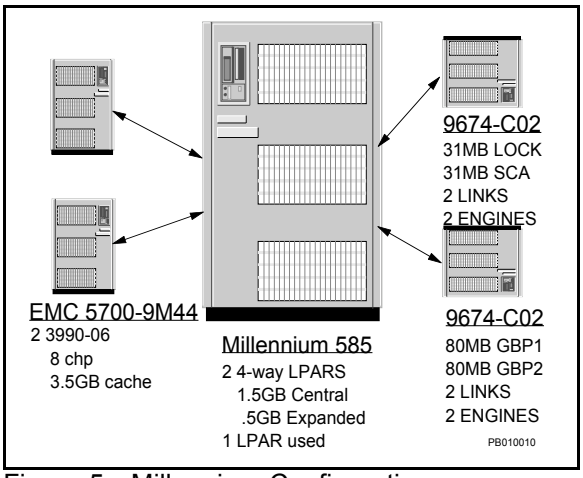

Figure 5 - Millennium Configuration

The credit checking application was tested on one LPAR of the Amdahl GS585, as shown in Figure 5. The GS585 was divided into two LPARs; each with four dedicated CPUs, 1.5 gigabytes of central storage and .5 gigabytes of expanded storage. Application and system datasets were housed on an EMC 5700-9M44, which was serviced by eight ESCON connections.

MVS/ESA SP5.2.2 was used as the supporting operating system, with DB2 Version 4 (PDO9642), CICS/ESA Version 4 (PDO9710), and CICSPlex/SM Version 1 Release 2 (PDO9709) used to support the application. Seven key functions of the credit checking application were chosen for the exercise. These functions were spread across four CICS/DB2 transactions. Table 1 profiles the transactions and their distribution.

TRAN.	SQL CALLS WKLD.	GET PAGES	% OF
ZMEN	10	30	49.5
ZWIN	18	30	40.9
ZSEC	33	72	3.4
ZHIS	13	19	6.2

Table 1 - Transaction Profiles

Selects	8.13	Get pages	31.35
Inserts	1.20	Sync reads	2.27
Updates	1.23	Buffer updates	4.10
Opens/closes	1.91	Pages written	1.26
Fetches	2.56	Async Writes	.317

Table 2 - Transaction Statistics

Although this is a CICS/DB2 implementation, few DB2 specific operations (like sequential prefetch, list prefetch, etc.) are invoked, primarily because DB2 was used to replace VSAM as the access method. Table 2 lists overall DB2 statistics per transaction.

At the start of each transaction, a SELECT is issued against an LTERM table, which keeps track of the last operation issued for a particular user. At the end of the transaction, the LTERM table is updated to reflect the outcome of the transaction. As we will see later, processing of this LTERM table proved to be a significant contributor to the increase in CPU per transaction for the two-way DB2 datasharing test.

STEP ONE - SINGLE CICS REGION, NO CICSPLEX, NO DATASHARING

Another key objective of this project was to test the application in its current state using newer releases of CICS and DB2. The existing application used CICS Version 3 and DB2 Version 3, so the first step was to rebuild the application using the new subsystems. This was a relatively simple task, just updating JCL to the new releases. The application build consisted of 5 major steps: 1) assemble and link the BMS maps, 2) assemble and link ASSEMBLER language routines, 3) translate CICS routines used as COPYLIB members, 4) SQL precompile, CICS translate, COBOL compile and link online modules, and 5) DB2 bind the packages and plan for execution.

The BMS maps were assembled in step 1after updating the SYSLIB dataset on the assembler step to point to the new CICS macro library (DFHMAC). Since the ASSEMBLER language routines did not reference CICS or DB2 libraries, no changes were needed to the JCL for step 2. The CICS COPYLIB members were translated in step 3 after updating the STEPLIB dataset to point at the new CICS DFHLOAD dataset. The online modules were produced in step 4 after updating the SQL precompiler STEPLIB file to point at the new DSNLOAD dataset, the CICS translator STEPLIB file to point at the new DFHLOAD dataset, and the link SYSLIB file to point at the new DFHLOAD. Finally, the packages and application were bound after updating the STEPLIB file to point at the new DSNLOAD dataset.

Since there were no changes required for the CICS system and application definitions for the single region implementation, a copy of the old CSD was obtained using DFHCSDUP, the CICS CSD update utility. Later experiments involved updating CSD definitions for transaction routing, CICSPlex processing, and session/link/network information, so the CSD extract program supplied as part of the CICS documentation was used. This program reads a CSD and produces the corresponding RDO statements necessary to define the CICS system.

Figure 6 shows the JCL and linkage editor control statements required to prepare the extract program for use.

```
//DOIT    EXEC COB2UCL
//COB2.SYSIN DD ……DFHSAMP(DFH0CBDC)
//LKED.SYSLIB DD……COB2.COB2LIB
//LKED.SYSLIN DD
//LKED.SYSLIN DD *
CHANGE EXITEP(BDEFCSD)
INCLUDE CICSLIB(DFHEXCI)
INCLUDE SYSLIB(ILBOSRV)
INCLUDE SYSLIB(ILBOCMM)
INCLUDE SYSLIB(ILBOBEG)
NAME DFH0CBDC(R)
```

Figure 6 - CSD Extract Preparation

One other slight change involved a new way to control the number of active tasks for a particular CICS transaction. In CICS 3.3 a combination of CMXT and CMXTLIM was used to control the number of tasks started for a particular transaction class. For CICS Version 4, CMXT and CMXTLIM are no longer used but a similar control can be accomplished via the TRANCLASS parameter. This identifies one of ten transaction classes (DFHTCL01-10) to which individual transactions can be assigned. The MAXACTIVE value, which can be set via a CEMT operator command, controls the number of active transactions for a particular TRANCLASS. For these experiments, the MAXACTIVE value was increased from 5 to 20.

The basic runtime methodology was as follows. Start DB2, reset application tables, archive the DB2 log to prevent an archive from occurring during the measurement, and shut down DB2. Restart DB2, start CICS, start the CICS/DB2 attachment facility, assure that DB2 and CICS are recording performance data at one-minute intervals, start TPNS to log on 1,000 users. After all 1,000 users have logged on to CICS and the application, start releasing users gradually until all 1,000 are active and issuing transactions (approximately 15 minutes). When all users are logged on and a steady state is reached, let the measurement run for an additional 60 minutes. At some point in the measurement interval (approximately 40 minutes into the steady state) the DB2 accounting trace is turned on for 3 minutes to collect application/transaction statistics. The workload is shutdown by quiesceing the users, stopping TPNS, stopping the CICS/DB2 attachment facility, stopping CICS and finally stopping DB2.
Since transactions ZMEN and ZWIN were issued over 90 percent of the time (see Table 1), they were each assigned a minimum of 10 and a maximum of 30 entry level threads in the RCT. Transactions ZSEC and ZHIS, which accounted for 3.4% and 6.2% of the workload respectively, were assigned to one of 60 pool threads. TOKENI=YES was specified to accurately collect CICS/DB2 transaction data.
Early on in the initial measurements, it seemed that DB2 was doing significantly more logging than was expected, based upon the number of log archive requests. In an effort to reduce logging, the Vertical Deferred Write Queue Threshold (VDWQT) was increased from the default of 10% to 50%. This parameter, which is new with DB2 Version 4, controls the flushing of buffers to DASD for a single DB2 application dataset. If one dataset has more than the default flush limit (10%) of the available pages in the buffer pool, then that dataset's pages are flushed to DASD. At this point I am guessing that the flushed pages also get logged, but this is only a guess. After increasing the VDWQT value to 50%, we were at least able to go 75 minutes without a log archive, which is what we were striving for.

Table 3 lists overall performance data for the single region implementation. The capture ratio of almost 90% is typical of DB2 workloads. It has been the author's experience that as the percentage of DB2 processing increases, so does the capture ratio.

TPNS Response Time	.28
TPNS Responses/minute	2631
Total Commits(ETR)	237,068
CPU Busy	60.00%
ITR	395,113
Capture Ratio	89.46%
Per Commit	
CPU seconds	.0344
locks	19.69
unlocks	1.55
get pages	31.35
sync reads	2.27
buffer updates 4.10	
pages written	1.26
async writes	.31
Average used	
central storage 640MB	
CSA<16M	660KB
CSA>16M	17.4MB
SQA<16M	707K
SQA>16M	13.3MB

Table 3 - Overall Statistics

Table 4 lists individual transaction statistics, compiled from the DB2 accounting records.

Transaction	ZMEN	ZWIN	ZSEC	ZHIS
#Samples	3357	2841	248	514
Class 1 Elapsed “	.2775	.3270	.1602	.0707
Class 1 TCB”	.0152	.0167	.0293	.0119
Class 2 Elapsed”	.0321	.0247	.0396	.0204
Class 2 TCB”	.0133	.0140	.0249	.0101
Selects	6	11	13	7
Inserts	0	2	4	0
Updates	1	1	2	1
Open/Close	1	2	4	1
Fetches	1	3	6	1
BP1 Get Pages	10	12	17	7
BP2 Get Pages	24	22	42	12

Table 4 - Individual Transaction Statistics

The I/O rate for this experiment was approximately 866 I/Os a second. Of this total, 343 I/Os a second were to DASD. Table 4A lists some CACHE statistics for the EMC DASD containing the DB2 system and application datasets.

CU	Read Rate	Hit Ratio	Write Rate	Hit Ratio	Read %	DASD Resp.
3220	7.1	.988	0	1.00	100	4ms
3260	77.5	.879	23.4	1.00	76.9	4ms
32A0	4.2	.999	4.0	.894	51.1	4ms
3300	47.1	.869	23.0	1.00	67.2	3ms
3360	8.5	.919	18	.999	32.1	3ms
33A0	4.4	1.00	4.1	.761	51.6	3ms

Table 4A – DASD Cache Statistics

The response times are very good, between 3 and 4 milliseconds across all the devices. These values should be taken with a grain of salt for a couple of reasons. First, this is the only application using a tiny portion of the hardware resources available on the 5700, and second, only a small portion of the live data used for this test was actually part of the TPNS script and transaction generation. Even though the data was being constantly paged in and out of the DB2 buffer pools, most of the I/Os were handled via the DASD subsystem’s cache, as shown by the high hit ratios. In a true production environment, where all the data is accessible to the application, more I/Os will have to be satisfied by going to the device, which will increase average DASD I/O response time.

	SYS1	SYS2	SYS3	SYS4	SYS5	SYS6
SYSGRS 0	0	541K	541K	0	1.1M	
SRVDC	10K	8K	7K	7K	15K	24K
SYSMCS 1434	463	79	100	9K	11K	
SYSWLM 879	879	879	879	879	4.5K	

Table 4B - XCF CTC Activity

The remainder of the I/O activity (approximately 523 I/Os a second) can be attributed to Sysplex communication. Table 4B lists the communication activity among the system under test and the other five members of the Sysplex. Each image in the Sysplex communicated with each other via four dedicated CTC connections, providing plenty of bandwidth for processing inter-system requests. The vast majority of the requests are for GRS processing among the three systems sharing access to the EMC DASD. During the measurement interval, the EMC DASD was in use only by the system under test.

STEP 2 - TOR/AOR, NO CICSPLEX, NO DATASHARING

Upgrading to a multiple CICS region setup using a Terminal Owning Region (TOR) and an Application Owning Region (AOR) required some updates to the CSD. Specifically, all files and transactions were defined as remotely existing in the AOR. For this stage, the transaction routing information is hard coded; no dynamic transaction routing programs are involved. ATTACHSEC(IDENTIFY) was specified for the MRO CONNECTION between the TOR and the AOR. This is necessary because the application does some security checking based on the USERID of the person logged in at the terminal issuing the transaction.

If ATTACHSEC is not specified, it defaults to LOCAL. When this is the case, at connection time between the AOR and the TOR, all USERIDs from the TOR assume the value of the LINK. When access is attempted, the application (with the help of RACF) thwarts the attempt because it is looking for a valid USERID. ATTACHSEC(IDENTIFY) assures that the USERID used at logon to the TOR, is the same USERID passed along to the AOR. The CONNECTION names and SYSIDNT values also match for each region, which is a requirement for CICSPlex/SM.

Although not required for the basic TOR/AOR change, at this point it was decided to run some CICS utilities to assure that no intertransaction affinities exist. This was done in preparation for CICSPlex/SM processing.

"An intertransaction affinity is a relationship between transactions, of a specified duration, that requires them to be processed by the same AOR. For example, if you have a pseudo conversation made up of three separate transactions, and each transaction passes data to the next transaction in the sequence via temporary storage, you specify that all three transactions must be processed by the same AOR, and that this affinity lasts for the duration of the pseudo-conversation. (if you did not define this affinity to CICSPlex/SM, each transaction would be routed to a different AOR and would therefore be unable to access temporary-storage data left by the previous transaction.)"[1]

Intertransaction affinities do not prohibit an application from using CICSPlex/SM, they just limit some of the workload balancing efforts because multiple transactions must be processed in a single AOR. If all intertransaction affinities are removed, than theoretically any AOR can process any transaction, which makes load balancing efforts easier to successfully implement.

Geoff Apps, manager of the Enterprise Computing Center (ECC) did an excellent job of analyzing the application for affinities with the help of the CICS Transaction Affinities Utility. This utility can SCAN load modules for an initial view of possible affinities, and can also run interactively with the workload in question, to dynamically DETECT affinities. Although there are some affinities that the utility cannot detect, both the SCAN and DETECTOR runs did not find any, at least not in the code paths we tested.

TPNS Response Time	.15
TPNS Responses/minute	2648
Total Commits(ETR)	239,938
CPU Busy	62.10%
ITR	385,504
Capture Ratio	90.33%
Per Commit	
CPU seconds	.0362
locks	19.72
unlocks	1.59
get pages	31.48
sync reads	2.27
buffer updates	4.10
pages written	1.26
async writes	.32
Average used	
central storage	661MB
CSA<16M	664KB
CSA>16M	17.5MB
SQA<16M	708K
SQA>16M	13.4MB

Table 5 - Overall Statistics

Overall statistics for this implementation are shown in Table 5. Average CPU seconds per transaction increased a little over 5% (.0344 versus .0362), with a drop in ITR of of 2.43%. Central storage and CSA use stayed about the same. Response time as recorded by TPNS decreased by .13 seconds, which could be looked at as a slight improvement, or could be looked at as A DRAMATIC RESPONSE TIME IMPROVEMENT OF ALMOST 50%!!!!! You pick.

Individual transaction statistics are shown in Table 6. The only significant difference here is the DB2 class 1 elapsed times for ZMEN and ZWIN, which increased slightly. CPU times per transaction remained basically the same as the base case.

Transaction	ZMEN	ZWIN	ZSEC	ZHIS
#Samples	3325	2879	264	491
Class 1 Elapsed "	.3442	.4311	.1707	.0774
Class 1 TCB"	.0146	.0171	.0292	.0118
Class 2 Elapsed"	.0360	.0321	.0502	.0266
Class 2 TCB"	.0124	.0142	.0251	.0101
Selects	6	9	13	7
Inserts	0	1	4	0
Updates	1	1	2	1
Open/Close	1	2	4	1
Fetches	1	3	6	1
BP1 Get Pages	7	10	18	7
BP2 Get Pages	24	19	42	12

Table 6 - Individual Transaction Statistics

STEP 3 - TOR/AOR, CICSPLEX, NO DATASHARING

Upgrading to the CICSPlex/SM environment required creating new tasks for the CICSPlex/SM operating environment, defining the CICSPlex/SM operating environment, updating existing CICS CSD definitions, updating existing CICS tables, and updating existing CICS startup parameters.

Two new tasks required for CICSPlex/SM are the Coordinating Address Space (CAS) and the CICSPlex SM Address Space (CMAS). The CAS is a started task, which provides a TSO interface for CICSPlex operation. The CMAS is a full blown CICS region used to manage the CICSPlex. "The CMAS implements the monitoring, real-time analysis, workload management, and operations functions of CICSPlex/SM, and maintains configuration information about the CICSplexes it is managing. It also holds information about its own links with other CMASs. In short, the CMAS is responsible for the single system image that CICSPlex/SM presents to the operator."[2]
If an existing CICS region is modified for use as a CMAS, assure that the guidelines for updating CMAS SIT parameters are followed, in particular

[1] CICSPlex SM Concepts and Planning, p.43

[2] CICSPlex SM Concepts and Planning, p. 16

the MXT, DSA, and ICVR parameters. Specifying less than the recommended values for these parameters can prevent the CMAS from being fully operational, which in turn will keep the CICSPlex from being operational. This can be a particularly difficult problem to diagnose, as there are no dumps or indications that a problem exists, other than a timeout message (EYUEI0029E). This information is documented in CICSPlex/SM Setup and Administration (SC33-0784-02), pp. 69-72.

The intricacies involved in properly defining a CICSplex are complex and are beyond the scope of this paper (see CMG95, CICSPlex/SM ESA Implementation and Capacity Planning for more detailed information). On a very high level, a Workload Manager Specification (WLMSPEC) defines the workload. The two major parts defined in the WLMSPEC are the TOR (TOR scope) and its associated AORs (AOR scope). The AOR scope is identified when the WLMSPEC is defined. For this project the AOR scope identified two AORs as part of a CICS system group. The TOR for this workload was identified by ADDing its scope to the existing WLMSPEC definition.

Updates to the CSD included specifying DYNAMIC(YES) in the transaction definitions in the TOR, and including the CICSPlex RDO definitions. The CICSPlex definitions are provided for you, and can be incorporated into your CSD by upgrading the CSD (UPGRADE USING(EYU941G1)) and adding the CICSPlex group to your startup list (ADD GROUP(EYU120G1) LIST(CPTORA)). The CSD updates are available as CSD upgrade load modules in the CPSM120.SEYULOAD dataset.

Several CICS tables need to be modified before using CICSPlex, including the Destination Control Table (DCT), Journal Control Table (JCT), Program List Table (PLT) and System Recovery Table(SRT). These tables are documented in CICSPlex SM Setup and Administration (SC33-0784-02) - Volume 1, Chapter 4, "Setting up a CMAS", and Chapter 5, "Setting up a MAS".

Table 7 lists overall statistics for this configuration. Although dynamic transaction routing is in place, there's only one available AOR, so no real decision making is being done when trying to determine where to route a transaction. Overall CPU per transaction increases (+10.46%), and ITR drops (-6.67%), both comparisons made to the base case. Central storage use increased from 640 to 720 megabytes, up 12.5%. CSA usage also increased by 60K below the line and almost one megabyte above the line. There was not much difference between the base case and this environment for individual transaction statistics shown in Table 8.

TPNS Response Time	.28
TPNS Responses/minute	2640
Total Commits(ETR)	240,038
CPU Busy	65.10%
ITR	368,721
Capture Ratio	90.57%
Per Commit	
CPU seconds	.0380
locks	19.76
unlocks	1.57
get pages	31.59
sync reads	2.28
buffer updates	4.11
pages written	1.27
async writes	.33
Average used	
central storage	720MB
CSA<16M	700KB
CSA>16M	18.3MB
SQA<16M	712K
SQA>16M	13.5MB

Table 7 - Overall Statistics

Transaction	ZMEN	ZWIN	ZSEC	ZHIS
#Samples	3322	2889	243	487
Class 1 Elapsed "	.2561	.3706	.1811	.0768
Class 1 TCB"	.0144	.0170	.0296	.0119
Class 2 Elapsed"	.0364	.0237	.0384	.0197
Class 2 TCB"	.0125	.0142	.0256	.0102
Selects	6	9	13	7
Inserts	0	1	4	0
Updates	1	1	2	1
Open/Close	1	2	4	1
Fetches	1	3	6	1
BP1 Get Pages	7	11	20	7
BP2 Get Pages	24	19	42	12

Table 8 - Individual Transaction Statistics

STEP 4 - TOR/AOR/AOR CICSPLEX, NO DATASHARING

The next step involved making a second AOR available for use by the application. All of the definitions were already in place from Step 3, so it was just a matter of starting up the second AOR at the beginning of the measurement.

The same RCT defined in STEP 1 was used to start the attachment facility for the second AOR. CICSPlex/SM based its transaction routing decisions using the MVS Workload Manager(WLM) and Goal mode processing. In fact, all of the measurements for this project were run using the WLM in Goal mode. With two AORs, a real decision needs to be made concerning where a transaction should be routed to keep the workload in balance, and this extra processing time shows up as increased CPU per transaction (+28.48%) and a decrease in ITR (-21.24%, compared to the

base case), as shown in Table 9. Average used central storage increased slightly (+12 MB), and CSA above the line also increased by 1 MB compared to STEP 3.

TPNS Response Time	.17
TPNS Responses/minute	2643
Total Commits(ETR)	238,878
CPU Busy	76.77%
ITR	311,160
Capture Ratio	91.22%
Per Commit	
CPU seconds	.0442
locks	19.82
unlocks	1.59
get pages	32.03
sync reads	2.30
buffer updates 4.11	
pages written	1.29
async writes	.35
Average used	
central storage 732MB	
CSA<16M	696KB
CSA>16M	19.1MB
SQA<16M	740K
SQA>16M	13.7MB

Table 9 - Overall Statistics

Transaction	ZMEN	ZWIN	ZSEC	ZHIS
#Samples	3351	2863	227	505
Class 1 Elapsed “	.5421	.5758	.2181	.0941
Class 1 TCB”	.0164	.0198	.0370	.0139
Class 2 Elapsed”	.0791	.0428	.0755	.0339
Class 2 TCB”	.0140	.0163	.0317	.0117
Selects	6	9	13	8
Inserts	0	1	4	0
Updates	1	1	2	1
Open/Close	1	2	4	1
Fetches	1	3	6	1
BP1 Get Pages	7	10	26	7
BP2 Get Pages	24	19	41	12

Table 10 - Individual Transaction Statistics

Individual transaction CPU times also increased, as shown in Table 10. These times are taken from the DB2 accounting records, which are normally used to compute chargeback information. The increases range from 7.89% to 26.27% compared to the base case.

STEP 5 - TOR/AOR/AOR CICSPLEX, ONE-WAY DATASHARING

Updating the existing DB2 system for one way datasharing was not a very difficult process, mainly because all of the SYSPLEX infrastructure (CF, CF links, policy, CF structures, CTCs, IOCDS changes) was already in place.

Undoing datasharing is a little trickier, but it can be accomplished when done with care. Datasharing involves the use of the coupling facility, for locking to begin with, and then for DB2 data pages when a second member of the group is started.

TPNS Response Time	.26
TPNS Responses/minute	2634
Total Commits(ETR)	240,433
CPU Busy	78.50%
ITR	306,284
Capture Ratio	91.13%
Per Commit	
CPU seconds	.0454
locks	19.87
unlocks	1.57
get pages	32.16
sync reads	2.31
buffer updates 4.11	
pages written	1.54
async writes	.17
Average used	
central storage 765MB	
CSA<16M	710KB
CSA>16M	18.9MB
SQA<16M	740K
SQA>16M	13.7MB

Table 11 - Overall Statistics

Changes to CPU times per transaction and ITR were very slight compared to STEP 4, as shown in Table 11. Latch conflicts per transaction rose from approximately .015 to .080, not a lot of conflicts per transaction, but a significant increase compared to non-datasharing. Likewise the individual transaction CPU times and average used central storage increased slightly compared to STEP 4, shown in Table 12.

Transaction	ZMEN	ZWIN	ZSEC	ZHIS
#Samples	3350	2884	256	438
Class 1 Elapsed “	.4823	.5715	.1622	.0706
Class 1 TCB”	.0171	.0203	.0393	.0142
Class 2 Elapsed”	.0372	.0299	.0518	.0235
Class 2 TCB”	.0146	.0168	.0339	.0120
Selects	6	9	13	8
Inserts	0	1	4	0
Updates	1	1	2	1
Open/Close	1	2	4	1
Fetches	1	3	6	1
BP1 Get Pages	7	10	28	7
BP2 Get Pages	24	19	42	12

Table 12- Individual Statistics

Coupling facility lock activity is graphed in Figure 7. Lock service time averaged approximately 194 microseconds, the average rate was approximately 138 locks a second. This type of activity puts very little stress on the CPU resources of the coupling facility, which averaged 1.4% busy. This is why the line representing coupling facility busy is almost hidden behind the bottom of the graph in Figure 7.

The CICSPlex/SM dynamic transaction routing facility as implemented in these experiments based its routing decisions solely upon load balancing between the two AORs. No consideration was given to things like trying to keep individual users and their data associated with a particular AOR. In these experiments all 1,000 users were active on both AORs at some time during the measurement. If users are bouncing back and forth between AORs, there's a good chance the user data is also bouncing back and forth between AORs.

Activity of this sort will show up in the DB2 statistics, namely data sharing locking and group buffer pool activity. Since only one member of the DB2 datasharing group was active for this step, activity to the group buffer pool was non-existent, with approximately 2.3 locks and 1.26 unlocks propagated to XES (global lock structure) per transaction. The next section with two active members of the datasharing group illustrates how application design affects DB2 datasharing performance.

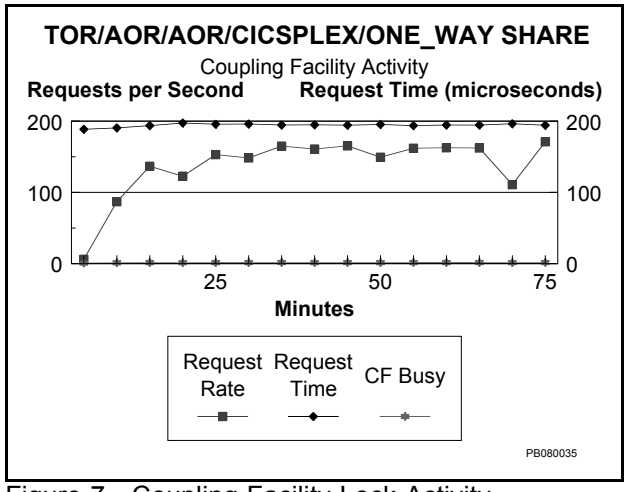

Figure 7 - Coupling Facility Lock Activity

STEP 6 - TOR/AOR/AOR CICSPLEX, TWO-WAY DATASHARING

Two way datasharing involved relating each AOR to a specific member of the DB2 datasharing group. This was accomplished by coding another RCT, similar to the original, but with a different DB2 subsystem ID (SUBID). The two AORs were then connected to two different members of the datasharing group.

TPNS Response Time	4.14
TPNS Responses/minute	2185
Total Commits(ETR)	204,460
CPU Busy	94.20%
ITR	217,048
Capture Ratio	91.63%
Per Commit	
CPU seconds	.0665
locks	22.16
unlocks	3.32
get pages	32.50
sync reads	3.03
buffer updates 4.15	
pages written	1.39
async writes	.28
Average used	
central storage 1007MB	
CSA<16M	809KB
CSA>16M	22MB
SQA<16M	744KB
SQA>16M	14.2MB

Table 14 - Overall Statistics

Changes to CPU time per transaction and ITR were substantial for two-way datasharing, as shown in Table 14. CPU time increased by 46% from .0454 to .0665 seconds per transaction, and ITR decreased by 29% from 306,284 to 217,048 transactions, compared to one-way datasharing. Most of the change in CPU was due to substantial increases in IRLM time per transaction. Notice average CPU busy of over 94%. CPU busy averaged close to 99% during the steady state portion of the workload, explaining the drop in ETR and increase in response time to 4 seconds. Compared to the base case, CPU per transaction increased by 93%, and ITR decreased by 45%.

Transaction	ZMEN	ZWIN	ZSEC	ZHIS
#Samples	2872	2570	200	369
Class 1 Elapsed "	.6729	.8440	.4387	.1953
Class 1 TCB"	.0216	.0263	.0518	.0185
Class 2 Elapsed"	.0861	.0818	.1464	.0611
Class 2 TCB"	.0192	.0228	.0458	.0161
Selects	6	9	13	8
Inserts	0	1	4	0
Updates	1	1	2	1
Open/Close	1	2	4	1
Fetches	1	3	6	2
BP1 Get Pages	7	11	30	7
BP2 Get Pages	23	19	42	12

Table 15 - Individual Transaction Statistics

Average used central storage increased by 32% from 765 to 1,007 megabytes, CSA below the line increased by 1 megabyte, above the line by 3 megabytes compared to one-way datasharing. The majority of the storage increase was due to the second member of the DB2 datasharing group.

Compared to the base case average storage use increased by 57% from 640 to 1,007 megabytes.

CPU times for individual transaction also increased substantially as shown in Table 15, with an average increase of approximately 29%.

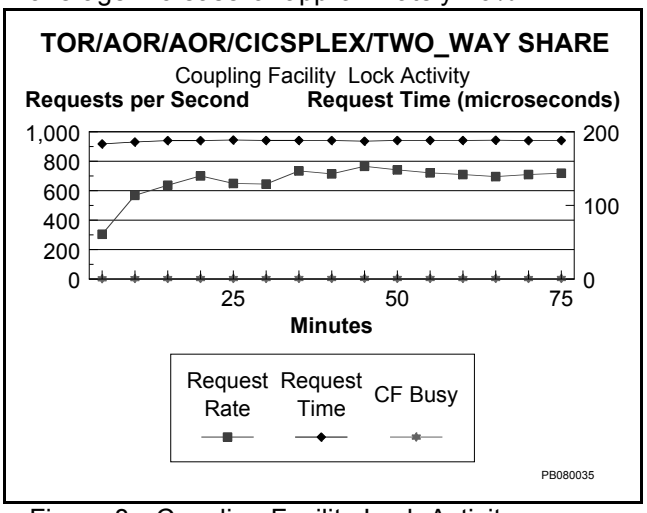

Figure 8 - Coupling Facility Lock Activity

Figure 8 graphs lock requests to the coupling facility, which averaged 674 locks per second and 188 microseconds response time. This is almost five times the locking rate from the one-way datasharing experiment. Coupling facility busy rises to an average of 3.7%, still plenty of capacity for growth.

The group buffer pools were contained in a separate coupling facility, its activity is shown in Figure 9. The coupling facility averaged 4.2% busy for the interval. On the average 435 requests per second were made to the group buffer pools, with response time of approximately 290 microseconds.

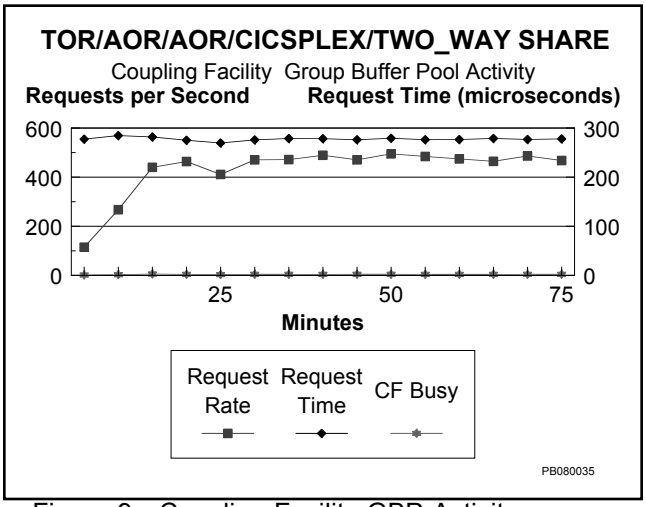

Figure 9 - Coupling Facility GBP Activity

Compare this response time with that of an I/O operation. Going all the way to the device for a random request can cost anywhere between 10 and 15 milliseconds, or 10,000 to 15,000 microseconds. Even if the data is cached at the device or control unit or both, response time will still be in the 1 to 2 millisecond range, which is between 3 and 7 times more than the response from the coupling facility. The point is that any perceived response time problems for coupling facility structures are still orders of magnitude better than the corresponding DASD response times.

Table 16 contains global locking and buffer pool data. Two descriptions are given for each field. The first is the title of the field from a DB2PM report. The second is the name of the corresponding source field in the DSECT describing DB2 statistics as mapped by the SMF100 record. The first two counters are the number of logical locks (22.16) and unlocks (3.32) per transaction. These numbers are slightly higher than the average locks(19.87) and unlocks(1.57) for all the other non-datasharing configurations.

QTXALOCK	Lock requests	22.16
QTXAUNLK	Unlock requests	3.32
QTGSLPLK	Lock requests (P-Locks)	1.514
QTGSUPLK	Unlock Requests(P-LOCKS)	1.496
QTGSLSLM 5.663	Synch.XES-Lock requests	
QTGSUSLM	Synch.XES-Unlock requests	2.983
QTGSCSLM	Synch.XES-Change requests	1.224
QBGLXD	Syn.Reads(XI)-data returned	.939
QBGLXR	Syn.Reads(XI)-R/W interest	.057
QBGLMD	Syn.Reads(NF)-data returned	.639
QBGLMR	Syn.Reads(NF)-R/W interest	1.848
QBGLSW	Changed pages sync.written	3.539
QBGLRC	Pages castout	2.067
QTGSIGLO	Suspends-IRLM global contention	.107
QTGSSGLO	Suspends-XES global contention	1.327
QTGSFLSE	Suspends-False contention	.017

Table 16 - Global Locking and Buffer Pool Activity

A portion of the logical locking and unlocking gets translated into global lock activity, which is displayed in the next five fields. P-Locks are used by DB2 to serialize update activity from multiple DB2 subsystems to a single row (data page, block, control-interval). Approximately 1.5 PLOCK lock and unlock requests are issued per transaction.

The next three counters are the number of synchronous lock requests to XES per transaction. Approximately 10 locks per transaction are propagated to XES, in the form of lock, change or unlock requests.

The next six counters detail activity to and from the group buffer pools, again on a per commit (transaction) basis. QBGLXD is the number of reads from the GBP to a local BP because of cross buffer invalidation, that is one member updates a page which exists in the other members local buffer pool. QBGLXR is the number of pages read from DASD to the local BP and registered in the GBP due to a member showing read/write interest in the page. QBGLMD is the number of reads satisfied from the GBP to a local BP. QBGLMR is the number of reads satisfied from DASD, while registering the page in the GBP. QBGLXR is the number of pages synchronously written to the GBP, and QBGLRC is the number of pages castout from the group buffer pool to DASD.

"The following three fields give the counts for the number of suspends due to global contention for lock requests. A global contention occurs when intersystem communication is required to resolve the lock conflict"[3] These three fields tell an awful lot about how much it will cost your application to share data. Suspends due to IRLM global contention are very low, about one every ten transactions. Suspends due to false contention are almost non-existent, about one every 50 transactions. Suspends due to XES global contention are very high however, averaging 1.33 per transaction. This is where the majority of the extra CPU needed for data sharing gets consumed.

How does one isolate the source(s) of the global contention? One method or approach is to turn on the lock trace by issuing the START TRACE command illustrated below.

-START TRACE(PERFM) CLASS(6) DEST(SMF)

This will record IFCID 0044 DB2 performance records to the SMF datasets. This IFCID describes lock suspend and identify requests to the IRLM. Requests are identified by DBID and OBID, and normally travel in pairs identifying the tablespace and table in question. The trace was turned on for a four minute interval. In that four minute interval, 3,768 global suspensions due to XES occurred. Of those 3,768 suspensions, 61% were locked out due to the LTERM table described at the beginning of this document.

This table is used by the application to keep track of status for a particular user, and is updated after every commit. Because CICSPlex makes transaction routing decisions based on capacity and not on location of data, a single user may be bounced back and forth between AORs, and their associated LTERM data would also be along for the ride, not only between AORs but between different members of the DB2 datasharing group. Refer back to Table 16, which indicates almost one cross buffer invalidation per transaction. The LTERM table is most likely the cause of the invalidation, because the user has been switched from one AOR/DB2 combination to the other AOR/DB2 combination.

[3] QTGS section from DSNDQWST expansion, DSN410.SDSNMACS

CONCLUSIONS

This application is at the high end of the spectrum in terms of datasharing overhead, with the main reason for this being the application design. The final full blown CICSPlex DB2 datasharing environment consumes almost twice the CPU power of the initial single region no datasharing design.

At first glance this may seem like a stiff price to pay. But when you consider the added flexibilty provided by datasharing, the automated load balancing provided by CICSPlex and the MVS workload manager, and the automated recovery and failover procedures provided by MVS and ARM, maybe it is worth the price, especially in environments approaching true 24X7 operations. Another point to consider is the scalability of the final setup. Adding more images in the Sysplex, and the associated AORs and DB2 datasharing members, will cause almost immeasurable differences in CPU usage, but will substantially expand the application's capacity for performance. The big CPU hit occurs when a second DB2 member is activated in the datasharing group. Activating subsequent members has very little effect on application resource consumption.

We ran out of time before we had a chance to use the second LPAR of the test machine. It would have been nice to see the ultimate capacity of all eight engines by starting up one of the AORs and DB2s in the second LPAR. And it would have been that easy, issue the start commands from the other MVS console. No job, JCL, or task changes are necessary. CICS is smart enough to use the coupling facility for communications between a TOR and an AOR in different images of the same Sysplex.

The flexibility offered with MVS Sysplex, CICSPlex, and DB2Plex is significant as shown in this paper. The use of each facility should be studied in relation to the business demands of the application. This paper shows that increasing resources can be accomplished with little or no application change.

ACKNOWLEDGEMENTS

A lot of people worked really hard to pull off this project, right in the middle of corporate re-shuffling at both the vendor's and customer's sites. At one particular low-point, when we were struggling to get two way datasharing working, I ventured downstairs to Sysprog row to ask some more favors of the CICS and MVS support people. Michael Wong and Rich Baker had already bent over backwards to provide the support we needed, and I was running out of ways to motivate them for still more help.

It turns out that both Michael and Rich are avid music fans, as is the author. I gave them both copies of *Christmas at the Mission*, a CD recently recorded by The Ohlone Chamber Singers (www.chambersingers.org), led by my friend and fellow musician Dennis Keller. I was hoping that the gift of music would help make their tasks a little more palatable.

We eventually finished the project, and after presenting the results, Rich came up and started extolling the fine job we had done. I was getting a little embarrassed at all the praise, after all, the presentation was in black and white, and I hadn't even told any good jokes. Turns out he was talking about the CD.

The next time you're stuck trying to implement a particularly complicated system or application, consider adding music to the mips and the memory. The ROI can be truly amazing.

.