DB2 Type II Index Processing in a Data Sharing Environment

William J. Raymond
David J. Young
Amdahl Corporation

The N-way sharing of DB2 data bases, which became possible with general release of DB2 Version 4, is one of the architectural foundations required to fully exploit Parallel Sysplex. Implementing this architecture is a complex, multifaceted process. Fundamental yet incremental improvements are required, such as those provided by the new Type II index facility for DB2 Version 4. The Type II index facility opens the door to other DB2 Version 4 enhancements, including parallel query processing, improved partition independence, row-level locking, and read-through locks. However, as you might expect, there are some tradeoffs.

1. INTRODUCTION

Type I indexes work the same in DB2 Version 4 as in all prior DB2 releases, so existing applications that use Type I indexes will work and perform as they always have. Staying with a Type I index however will prevent the application from using most of the data sharing and parallel processing built into Version 4. According to IBM, "None of the functions of type I indexes have been removed with this release. However no new functions are planned for type I indexes, and support for them might be removed in future releases."[1]

One of the fundamental improvements of Type II index processing is the elimination of locks on index pages and subpages. This reduces contention and the probability of deadlocks, which can lead to higher levels of concurrency. Type II indexes also permit the reading of uncommitted data (read through locks, or "dirty reads"), and improved partition independence, allowing concurrent access to logical partitions of a non-partitioned index.

Type II indexes also use a smarter approach than Type I when it comes to dealing with non-unique keys. DB2 uses a Record Identifier (RID) to point from the index to the desired data. With a Type I index, the RIDs for duplicate keys are not kept in any particular order. If the key field changes, DB2 must do a sequential search through all the RIDs for that key value, until it finds the desired RID, and then remove it from the twin chain. This search can consume vast amount of CPU and elapsed time, depending on the number of duplicate keys, and the location of the particular RID in the twin chain.
A Type II index maintains the RIDs in order so that a binary search can be used to locate the RID in question. This can drastically improve the CPU and elapsed time required for changing the value of a duplicate key, especially when there are thousands of duplicates or more.

The extra functionality of Type II indexes and DB2 Version 4 comes at a price, however. Row-level locking can, in some cases, significantly increase resource utilization without any apparent benefit to the end user.

The following discussion investigates some of the tradeoffs involved in Type II index processing, and provides a high-level description of concepts and metrics for DB2 Version 4 data sharing. The approach was to conduct a series of experiments designed to measure some of the costs of Type II index processing. These experiments evaluate CPU requirements for Type II indexes, compare these with Type I, and yield some initial conclusions.

2. WORKLOAD DESCRIPTION

All experiments are based upon tables defined by the TPC-C workload. (TPC-C is a standard benchmark defined by the Transaction Processing Performance Council.) Because this is an initial investigation, a relatively small two-warehouse implementation was selected. The tables for this implementation were defined and loaded according to the TPC-C specification. Table 1 lists the DASD space requirements for both Type I and Type II indexes.

[1] *Release Guide*, p.45, SC26-3394-01

TPC-C Table	High Used RBA Type II	Type I	Percent Change
2 Warehouses	98K	98K	0
20 Districts	98K	98K	0
60K Customers	1.3M	1.1M	+18.55
60K Orders	1.3M	1.1M	+20.09
18K New Orders	401K	339K	+18.28
600K Order Lines	15.7M	13.6M	+15.26
100K Items	1.3M	.9M	+45.34
200K Stock	3.4M	2.7M	+25.26

Table 1 – DASD Index Space Utilization

As shown, there is an increase in space use for Type II indexes, ranging from 0 to 45.54 percent. The biggest difference occurs for the index built upon the Items table. The key for this table is a unique four-byte integer. At first glance, the RUNSTAT statistics for this index do not seem to be much different from those for the other indexes, except for a few fields.

Table 2 following lists some of the fields from SYSIBM.SYSINDEXES. Notice that the Item index from Table 1 has the greatest delta in space use when comparing Type I and Type II indexes. Notice also that the same index is the only one built on a single field, as indicated by the value of 100K for both Firstkeycard and Fullkeycard. Firstkeycard counts the number of unique values for the first field in an index, and Fullkeycard counts the number of unique values for the full key. One other difference is the Item index has only two levels, while most of the other indexes have three levels. Why the space increase is so big for this index is a mystery.

Index	Firstkey card	Fullkey card	Nleaf	Nlevels
WHIN01	2	2	1	2
DIIN01	2	20	1	2
CUIN01	2	60K	315	3
OIIN01	2	60K	315	3
NOIN01	2	18K	95	2
OLIN01	2	600K	3801	3
ITIN01	100K	100K	304	2
STIN01	2	200K	827	3

Table 2 – DB2 Index Runstat Statistics

Two DB2 subsystems were created to process the test application: DB1G and DB2G. The test application consisted of four batch jobs, two submitted to each DB2. Each job sequentially updated rows in the Order Line table by placing the current timestamp in one of the comment fields. In order to cause as much locking and contention as possible, the jobs submitted to DB1G updated the same rows as the jobs submitted to DB2G. The jobs for DB1G were submitted slightly ahead of the jobs for DB2G, and each job updated 50 rows per commit. A total of 3,000,000 updates were issued.

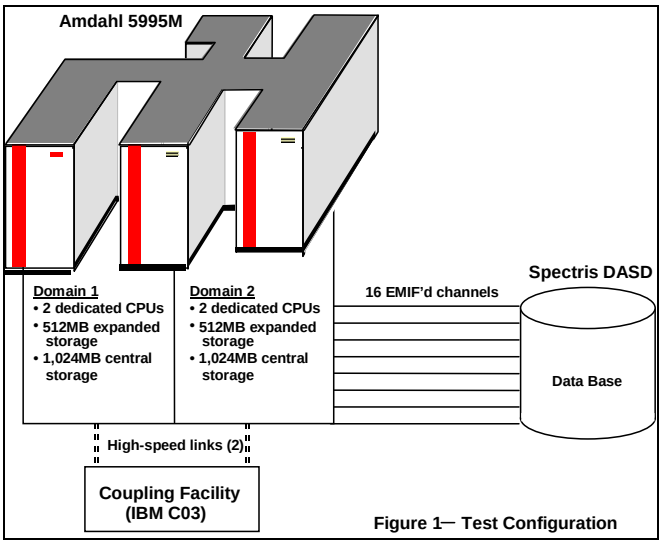

Figure 1 – Hardware Configuration

As shown in Figure 1, the host machine was an Amdahl 5995M, configured with two domains. Each of these domains consisted of two dedicated CPUs, 512MB of expanded and 1,024MB of central storage. The data base lived on Spectris DASD, connected to the M by 8 EMIF'd channels. OS/390 (MVS/ESA 5.2.2) was the operating system, and the applications were coded in VS COBOL II.

The coupling facility was an IBM CO3 serviced by two high-speed links, and configured with two processors and 1 gigabyte of memory. The suggested default values were used for the DB2 data structures on the coupling facility: 20MB for group buffer pool 0, 5MB for the lock structure, and 8MB for the SCA. The SCA is a list structure used to hold, among other things, damaged pages and information relating to their recovery. These structures are defined in the MVS Coupling Facility Resource Management (CFRM) policies.

One of the trickier steps to a successful data sharing implementation is getting the names right for the coupling facility data structures. The prefix for each structure name is the name chosen for the DB2 group during DB2 install. If you install via the SPF panels, it is parameter GROUP NAME on panel DSNTIPK, or GRPNAME on the DSNGGRP macro used to define the DB2 start up parms DSNZPARM. These names (i.e.DSN410_GBP0) must match the ones defined in the CFRM policy that defines the DB2 coupling facility data structures.

3. RESULTS

Three different index setups were used for these experiments: Type I index, Type II index with page locking, and Type II index with row locking. All three versions of the database loaded using the same amount of CPU (2.52') and with the same elapsed time. There were considerable differences in CPU and elapsed time for the 3,000,000 updates, as shown in Table 3.

	Type I	Type II Page	Type II Row
Elapsed	22'52"	16'19"	23'14"
Updates per Second	2,186	3,064	2,152
Appl. CPU	30'51"	13'2"	22'10"
DB2 System CPU			
MSTR TCB	9.61"	9.14"	8.11"
MSTR SRB	3'2"	3'1"	10'28"
DBM1 TCB	3"	3"	3"
DBM1 SRB	56"	57"	1'6"
IRLM TCB	1"	1"	1"
IRLM SRB	2'9"	2'16"	2'20
Total CPU	37'11"	19'29"	36'15"

Table 3 – Update Elapsed and CPU Times

For update activity as intense as this, exceeding 3,000 updates a second, it is important that the DB2 logs and application datasets are on devices like Spectris, which can tolerate the corresponding I/O rates. In these experiments, activity rates of 200 I/Os per second with average device response time between 3 and 4 milliseconds were not uncommon.

The results demonstrate a substantial reduction in total CPU time as well as elapsed time for the Type II page lock index when compared with the Type I index. CPU and elapsed time for the Type I and Type II row-lock indexes are almost equal. This is an example of an application which works much better with Type II page locks. The applications finish properly, with no deadlocks in either the Type I or Type II environment. Other environments with more stringent concurrency requirements may need to absorb the extra CPU required for row-level locking. Table 4 lists some of the locking statistics for Type I, Type II page, and Type II row environments.

There is a lot less locking (pardon the alliteration) in the Type II environment—and even less unlocking, due to the new and improved index design. The number of logical dataset reopens is slightly different in all three environments. This occurs when a not-in-use open dataset becomes in-use.

	Type I	Type II Page	Type II Row
Total Locks	6.07M	3.08M	3.24M
Total Unlocks	3.06M	69K	245K
Total Latch Suspends	1555	1957	4063
Total Logical Reopens	20K	14K	34K
Locks/Tran	101	51	54
Unlocks/Tran	51	1	4
Latch Suspends/Tran	.05	.06	.15
Logical Reopens/Tran	.33	.24	.57

Table 4 – Locking Activity

The cost of this activity may show up in the extra SRB time for the Type II row-lock setup.

IBM has provided new statistics that detail group buffer pool activity and XES interaction. Statistics are provided via the DISPLAY GROUPBUFFERPOOL command and in the DB2 Statistics trace record (SMF100). Some of these statistics are shown in Table 5.

	Type I	Type II Page	Type II Row
Read Data Returned	45K	46K	55K
Read Data Not Returned			
Dir. Entry Existed	46K	46K	47K
Dir. Entry Created	31K	31K	30K
Write Changed Pages	179K	179K	181K
Reclaim Data Entry	72K	73K	73K
Castouts	94K	95K	95K
XI Reclaims	0	0	0
XI Writes	91K	91K	102K
Read Hit %	37.4	37.7	42.07

Table 5 – DB2 Group Buffer Pool Statistics

The group buffer pool statistics are basically the same for all three environments. "Read Data Returned" is the number of successful reads to the group buffer pool. And under "Read Data Not Returned," Dir. Entry Existed means the page was not in the group buffer pool but was read from DASD. An existing global buffer pool directory entry for this page existed and was re-used. "A directory entry specifies the location and status of an image of a page somewhere in the data sharing group, whether the image appears in the group buffer pool or in one of the member buffer pools".[2] Dir. Entry Created means the page was not in the group buffer pool, but was read from DASD and an entry was created for it in the group buffer pool.

[2] *Data Sharing Planning and Administration*, p. 54, SC26-3269-01

"Reclaim Data Entry" counts the number of times an existing data entry was used. A high value for this number is acceptable, as long as there are few cross invalidations (XI) resulting from directory reclaims. In all three environments, there are zero XIs from directory reclaims.

"Castouts" is the number of pages written from the group buffer pool to DASD. This is the first indicator that perhaps the group buffer pool is too small. Two thresholds that control castout activity are class castout threshold and group buffer pool castout threshold. Class castout occurs when the number of pages updated for a certain class exceeds the specified percentage of the total group buffer pool. This value defaults to 10 percent. Group buffer pool castout threshold occurs when the total number of updated pages for all classes exceeds the specified percentage of the entire group buffer pool. This value defaults to 50 percent.

Due to the small size of the group buffer pool, and the intense update activity, the class castout threshold was reached an inordinate number of times. This caused the pages to be moved from the group buffer pool to the local buffer pool of the DB2 in charge of the castout process, where they are then written to DASD.

The "Read Hit %" is calculated by dividing the number of successful reads to the group buffer pool by the total number of reads. The read hit percentage for all three environments is respectable, between 37 and 42 percent. These ratios result from the nature of the applications: they update the same rows, with the jobs on DB1G just slightly ahead of the jobs on DB2G. In essence, DB1G primes the group buffer pool for DB2G.

Figure 2 graphs the coupling facility lock requests per minute for all three environments. As shown earlier in Table 4, locking activity for the Type II page-level index is substantially less than both Type 1 and Type II row-level indexes. The values for Figure 2 were taken from the Coupling Facility Usage Summary Report. The new RMF post processor, which provides the report, will not summarize across intervals for this information, but will only report on the interval boundary. So if you are recording data

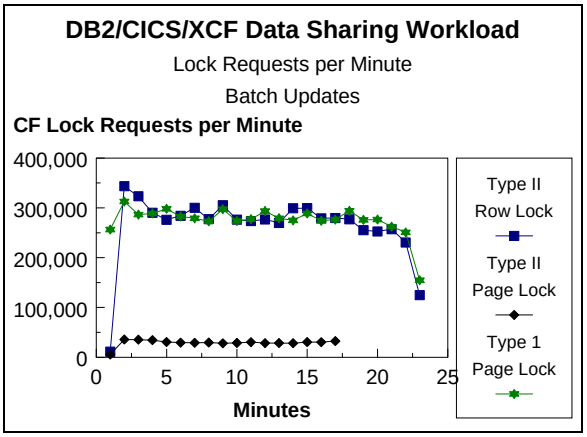

Figure 2 – Coupling Facility Lock Requests

in 1-minute intervals, you get Coupling Facility Usage Summaries in 1-minute intervals. (The new RMF post processor also provides the other reports discussed below.)

It is important to keep in mind that most lock requests are synchronous calls to the coupling facility. A synchronous request in many cases will consume CPU cycles until the request is completed. This extra CPU is accounted for in two places: DB2MSTR SRB time and application CPU time, as shown in Table 3.

More useful information from the Coupling Facility Usage Summary is shown in Table 6.

Struct. Name	Alloc Size	List/dir Entries Tot/Cur	Data Elem. Tot/Cur	Lock Entries Tot/Cur
SCA	8M	13K	25K	N/A
		144	188	N/A
LOCK1	5M	7357	0	2097K
		15	0	0
GBP0	20M	20K	4047	N/A
		12K	4043	N/A

Table 6 - Coupling Facility Usage Summary

The size in megabytes for each structure is listed under "Alloc Size." Space in the group buffer pool is occupied by two types of structures: (1) directory entries that contain information about a page, and (2) data elements, which hold an actual page of data. The default ratio of directory entries to data elements for the group buffer pool is five directory entry pages per data page. If this ratio is unacceptable, it can be altered via the ALTER GROUPBUFFERPOOL command.

Table 7 following contains statistics from the Coupling Facility Structure Activity Report. This report details the number of requests, their average service time in microseconds, and the number of delayed requests due to no available subchannel.

# Req Total	440K
# Req/sec	7,331
# Sync	440K
# Async	0
# Changed	0
# Req Deferred	5522
Contention	5522
False Contention	373

Table 7 – Coupling Facility Structure Activity

A false contention occurs when two separate locks hash to the same lock entry. A large number of these could indicate an insufficient number of locks. The number of locks in the structure can be increased by two methods: (1) increase the size of the structure in the CFRM policy, or (2) reduce the number specified for MAXUSRS on the IRLMPROC. This represents the expected number of DB2 members in the data sharing group. Specifying a value less than 8 allows the IRLMs to use lock entry sizes of two bytes. Specifying values from 8 – 23 causes four-byte lock entries to be created, and specifying values of 24 through 33 causes eight-byte entries to be created. Most installations can exist very well with the reduced lock space caused by specifying 7 or less for MAXUSRS. This in turn will minimize the number of false contentions caused by inadequate lock space. Make sure the value specified reflects your configuration.

For the particular 1-minute interval examined, 373 requests were false contentions out of a total of 438,000 requests. The number of false contentions is than 0.1% of the total requests; this value seems to indicate the size of the lock structure is adequate.

Activity to the DSN410_GBP0 group buffer pool is also reported in the Coupling Facility Activity Report. Included are data access statistics such as the number of reads, writes, castouts, and cross buffer invalidations. Figure 3 graphs the cache requests for all three environments. The number of requests for the Type II page-level index is higher and shorter than the other two environments as a result of the shorter elapsed application time.

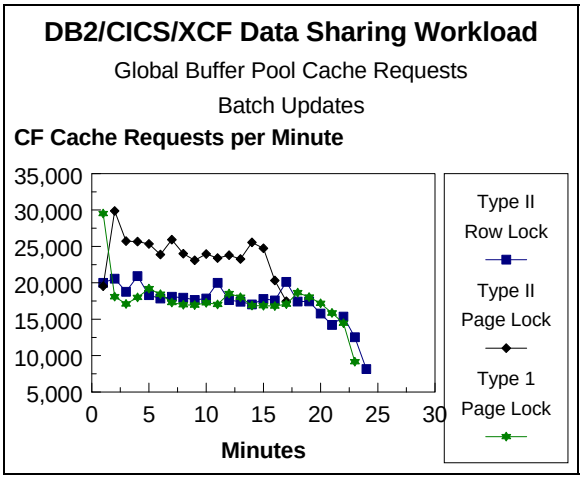

Figure 3 – Global Buffer Pool Cache Requests

With over 7,700 requests a second, you would expect the CPU activity on the coupling facility to be quite brisk. "Coupling Facility Busy" is reported on the Coupling Facility Usage Summary Report as AVERAGE CF UTILIZATION (% BUSY). This also tells you the number of processors defined for the CF, and the number in use (EFFECTIVE). These two numbers should be the same, which in this case is two IPs. Figure 4 graphs "Coupling Facility Busy" for the three environments.

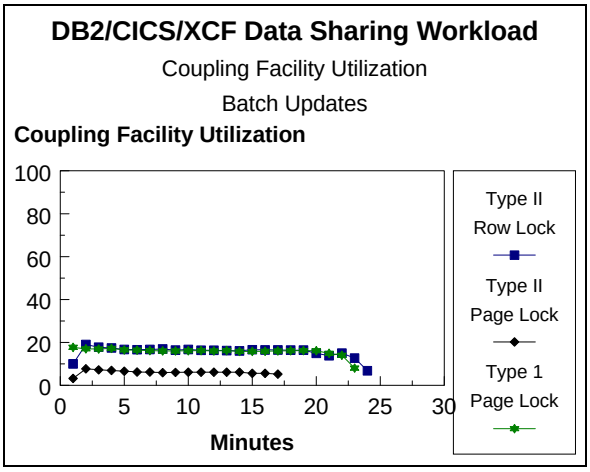

Figure 4 – Coupling Facility Utilization

Keep in mind that coupling facility utilization for the workloads described in this paper push the performance envelope to extremes: Not even the most advanced bleeding edge customer shops will experience this type of activity. In fact it is the primary author's experience that the vast majority of shops using data sharing in a full production workload have trouble getting the CPUs in the coupling facility busy more than five or six percent of the time.

Coupling facility utilization can exceed 70 percent with almost immeasurable differences in end user response times. This type of activity will drive the coupling facility links very hard, and evidence of this will show up in the Subchannel Activity Report. Some of this information is shown in Table 8 following.

# REQ		
TOTAL		--BUSY--
AVG/SEC	--CONFIG--	-COUNTS-
418869	SCH GEN 4	PTH 2
6981.1	SCH USE 4	SCH 32
	SCH MAX 4	
	PTH 2	

Table 8 – CF Subchannel Activity

In addition to providing link activity, the Subchannel Activity Report will tell you if the number of subchannels configured in the IOCDS (SCH GEN) is equal to the number of subchannels in use (SCH USE). SCH MAX shows the maximum number of subchannels available in the current pathing configuration. If the number of SCH BUSY COUNTS (requests queued) starts exceeding 10 percent of the total requests, and end user response time is degrading, then this could indicate an insufficient number of CF links. "Path busy" usually occurs only in a partitioned environment where multiple MVS images share the same CF links. However, this is a potentially dangerous situation because the processor will spin issuing requests until it obtains the path. Lots of "path busy" means lots of CPU spinning, which could mean a significant drop in effective MIPS for your installation. As shown in Table 8, only 32 requests out of 418,869 were queued, indicating a sufficient number of CF links.

One final indicator of the amount of processor utilization for each environment is shown in Figure 5, which graphs the Service Rate as reported by RMF in its summary report.

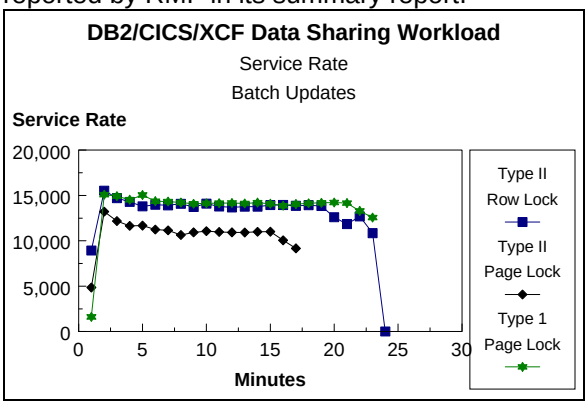

Figure 5 – SRM Service Rates

As expected, the amount of service consumed by the Type II page-lock index is less and shorter in duration than the other two environments.

4. CONCLUSIONS- BATCH PROCESSING

Using Type II page-level indexes can substantially reduce CPU and elapsed times for some long-running batch applications, especially those with intense update activity. However, the amount of space required for a Type II index can be substantially more than the same index implemented as Type I. Another drawback to data sharing is that existing Type I indexes must be converted to either SUBPAGE 1 or Type II indexes before they can participate in data sharing. This conversion is (at the least) a formidable and (at the most) an almost unwieldy task for large DB2 installations.

The new RMF post processor provides many useful statistics for analyzing coupling facility activity and planning for its capacity. This will be especially important in the next phase of workload implementation, which consists of implementing a CICSPlex environment on top of the existing DB2 subsystems. This setup will allow controlled execution, measurement, and analysis of various coupling facility and Parallel Sysplex configurations, including the MVS/ESA Workload Manager, with the ultimate goal of relating their performance to the metric that counts the most: user response time.

5. ADDING CICS & TPNS

The workload described in DB2 Type II Index Processing in a Datasharing Environment Has been enhanced to include TPNS, CICS/ESA, and CICSPlex SM/ESA as shown in Figure 6. Instead of batch jobs which update 50 rows per commit in a sequential nature, we now have individual CICS transactions, which will begin at a random point within the database, and update the next 50 rows.

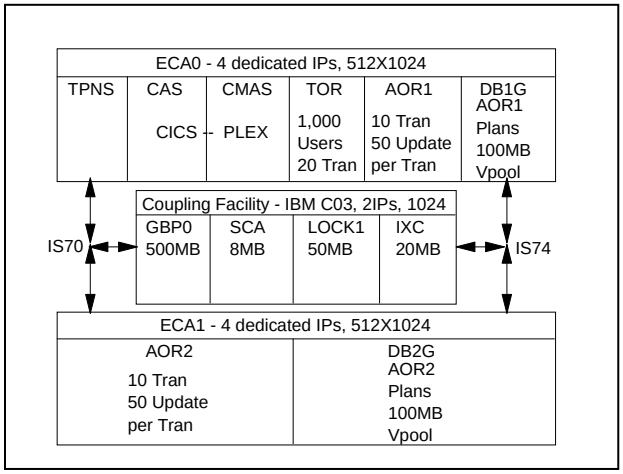

Figure 6 - CICS/DB2/XCF Environment.

TPNS is used to simulate the 1,000 users defined in the Terminal Owning Region (TOR). Twenty identical DB2 Update transactions, each updating 50 rows, are defined in the TOR: Ten transactions hard coded for Application Owning Region one (AOR1) and ten hard coded for AOR2. The transactions in AOR1 all access DB2 subsystem DB1G; likewise the transactions defined for AOR2 all access DB2 subsystem DB2G. The TOR communicates with the AORs via MRO: For local communications to AOR1, CICS uses MRO with the XM option, and for remote access across the sysplex CICS uses MRO with the XM/XCF option.

The configuration shown above was run on a Millennium, two dedicated domains, each configured with 512MB real, 1,024MB expanded, and four dedicated IP's. Access to the coupling facility was provided via two IS channels, shared between the two domains ECA0 and ECA1. Two dedicated IP's were used in the coupling facility, which was configured with 1,024 MB of real. The storage was allocated as follows: 500MB for DB2 group buffer pool 0, 8MB for the DB2 SCA, 50MB for the DB2 lock structure, and 20MB for the IXC_DEFAULT list structure. IXC_DEFAULT is used by the CICS TOR to communicate with AOR2, using the XCF option of MRO.

6. ROW LEVEL LOCKING

Unfortunately the workload in its current state doesn't seem to scale very well, due to an increasing number of XES lock requests per transaction. Lock requests which are propagated to XES are done so because a global conflict has occurred, one which cannot be solved by the local IRLM. What makes this problem so perplexing is that even though the two members of the DB2 group are sharing the same table, they are not sharing the same data. There should be few if any global conflicts.

One way to determine the source of locking conflicts is to turn on performance trace, class 6, which records lock suspensions(IFCID 44). This will identify the table space/table causing the conflicts. Unfortunately, even DB2 is not sure whether a global conflict truly exists, as indicated in the output produced for IFCID 44. One of the reason codes given for a lock suspension is IS (inter-system communication required), which according to the documentation indicates there may or may not be a conflict.

Figure 7 illustrates the classic diving ITR curve, one which can occur if the workload doesn't scale, or the processor doesn't scale. In this case it appears to be the workload.

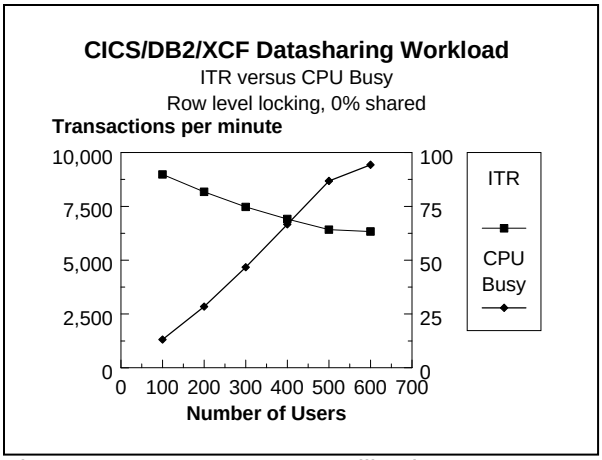

Figure 7 – ITR versus CPU Utilization

The ITR takes a dive due to the increasing number of XES locks per transaction, as shown in Figure 8.

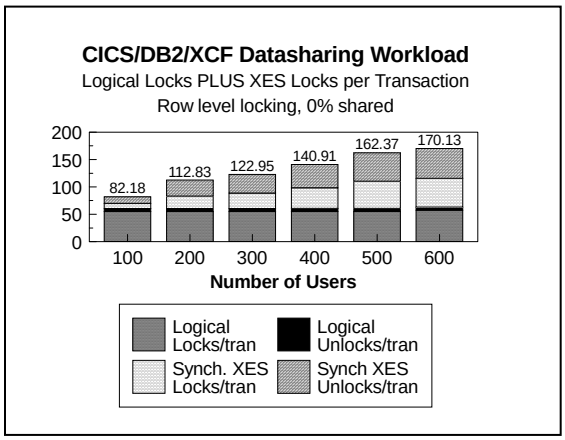

Figure 8 – XES and Logical Locks per Tran

Logical locks and unlocks per transaction are issued at the local DB2 level and remain consistent throughout the measurement period, approximately 55 locks and 5.5 unlocks per transaction. As the activity increases, the number of local locks/unlocks propagated to XES increases dramatically, causing service times to degrade for the DB2 lock and cache structures and improve for the CICS IXC-DEFAULT list structure.

Figure 9 graphs the transaction rate (ETR) versus response time.

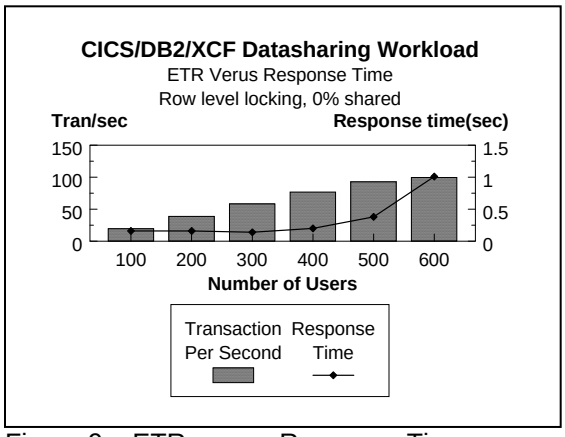

Figure 9 – ETR versus Response Time

In this setup maximum ETR is approximately 100 transactions per second, with an average response time of one second.

Figure 10 shows the lock activity and lock service time for the measured environments. The scale on the left is requests per second and corresponds with the bar graph, the scale on the right is service time in microseconds and corresponds to the line. At its busiest the Coupling Facility is processing almost 11,000 locks request per second, split almost evenly between ECA0 and ECA1. The service time per lock request varies from approximately 167 microsec. at 100 users to 222 microsec. at 600.

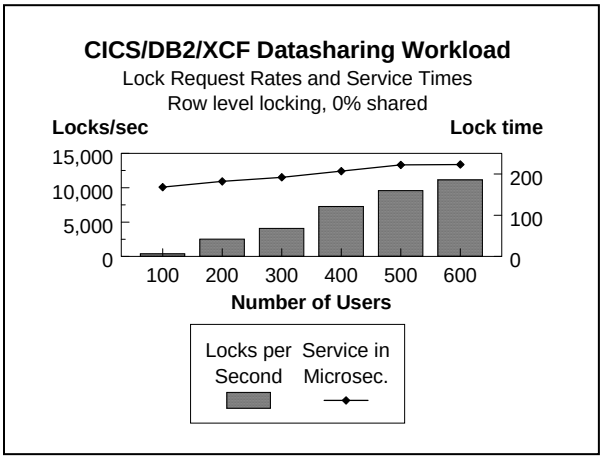

Figure 10 – XES Lock Request Rates

The increased XES lock activity also shows up in average CPU/ tran, as shown in Figure 11.

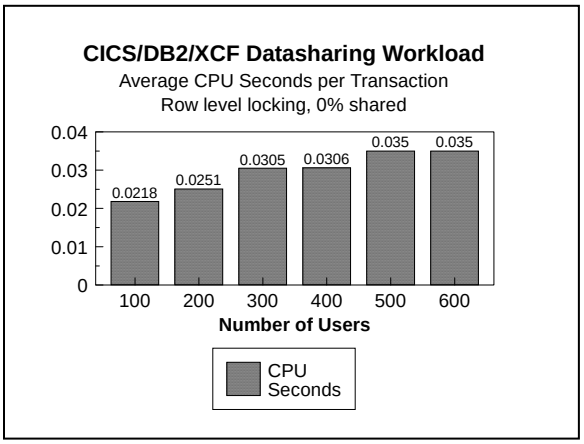

Figure 11 – CPU Seconds per Transaction

Compared to lock requests, there were relatively few cache requests for pages in the global buffer pool. This is illustrated in Figure 12.

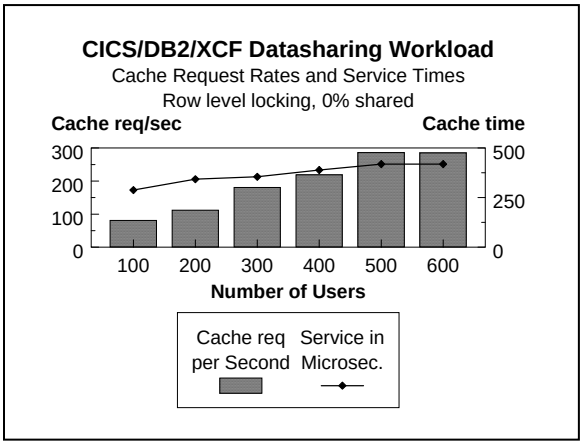

Figure 12 – Cache Rates and Service Times.

The maximum cache request rate is approximately 275 pages per second, with a service time ranging from 281 to 416 microseconds. For this measurement, there were a total of 690,904 accesses to the group buffer pool cache, with writing changed pages accounting for over 98% of the requests, as shown in Table 9.

XI Read data returned	3
XI Read no data returned	1
Read data returned	1
Read data not returned,	
Dir. Entry created	6,024
Appl. write changed page	673,149
Sys. write changed page	4,739
Other SES requests	1,017

Table 9 – XES Cache Activity

This was the only activity to the group buffer pool. If any of the data had been shared between the two DB2 systems, that is they both wanted to update the same page, rather than updating different pages on the same table, then there would be substantially more XI Reads. This is why it is so confusing to my feckless mind as to why the locks need to be propagated to XES.

Figure 13 shows the activity and service rates for the CICS MORO/XCF structure IXC_DEFAULT.

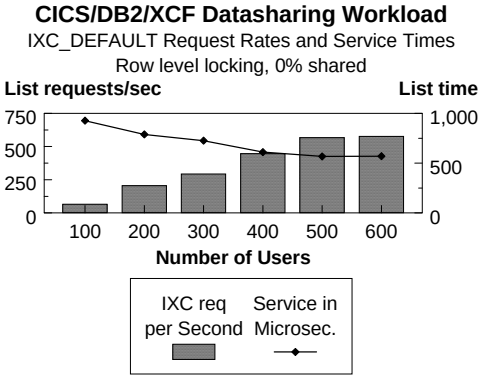

Figure 13 – IXC_DEFAULT Activity

This structure is used when the TOR on ECA0 needs to communicate with the AOR on ECA1. Half of the total transactions are routed across the sysplex via this mechanism. This is a list structure, accessed asynchronously, and its service time actually gets better as activity increases, from a high of 930 microseconds to a low of 600 microseconds.

7. PAGE LEVEL LOCKING

As shown in the batch version of this workload, locking activity to the coupling facility increases dramatically when the locksize for the tablespace is changed from PAGE to ROW. In an effort to make the workload scalable, locksize was set back to PAGE from ROW. This had the desired effect of reducing the XES locking activity, which in turn leads to a scalable workload. Unfortunately the activity rates to the coupling facility decreased too much to make this a usable CF tool. To increase CF activity, the amount of datasharing between the two DB2 subsystems was increased from zero percent to thirty percent. Figure 14 graphs ITR versus CPU busy for page level locking.

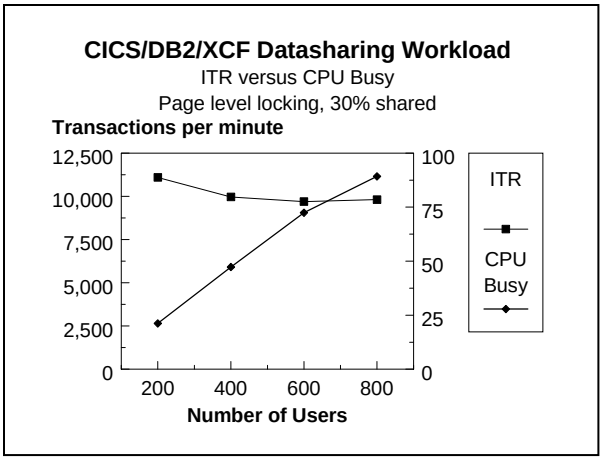

Figure 14 – ITR versus CPU Page Locking

The ITR line for page locking is much flatter than for row locking, decreasing 13% over the entire interval, and staying flat (0%) if only the last three intervals (CPU busy > 45%) are accounted for. Compare this with a drop in ITR of 42% and 18% for the corresponding intervals in Row level locking.

Workload consistency also shows up in CPU per transaction, as reported by the DB2 SMF Accounting records. Average CPU per transaction ranges from .0204 to .0225 seconds, as shown in Figure 15.

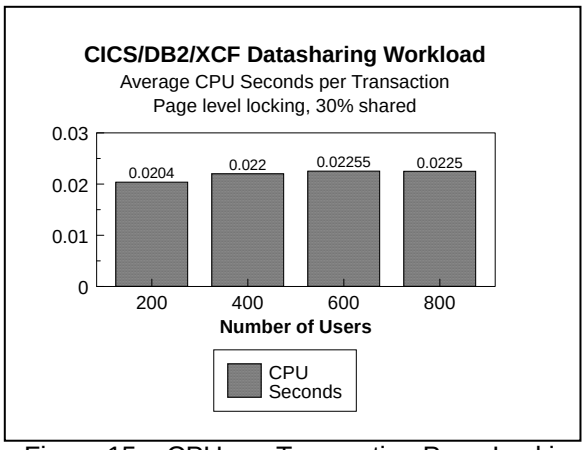

Figure 15 – CPU per Transaction Page Locking

Figure 16 graphs ETR versus transaction response time for page level locking.

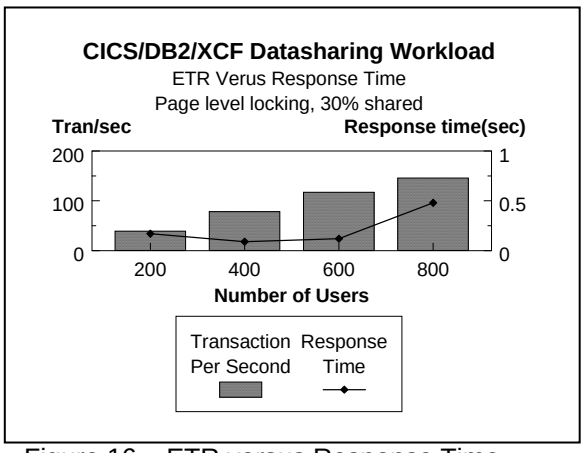

Figure 16 – ETR versus Response Time

The major reason application CPU per transaction stays consistent is XES lock activity per transaction stays consistent: It doesn't skyrocket as in the row level locking setup. This is shown in Figure 17.

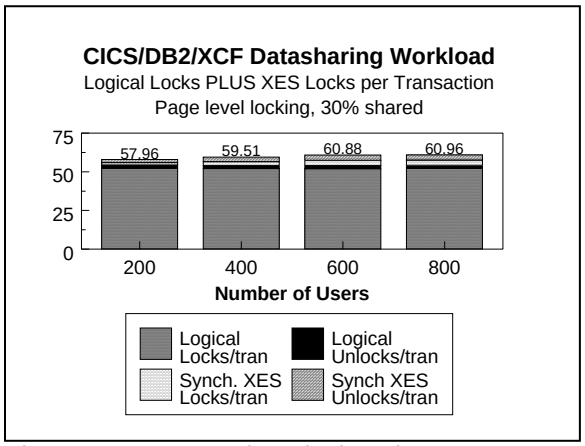

Figure 17 – XES and Logical Locks per Tran

XES lock activity ranges between 3 and 7 operations per transaction for page level locking. Compare this with between 24 and 106 XES lock operations for row level locking. Figure 18 shows XES lock activity for both DB2 systems.

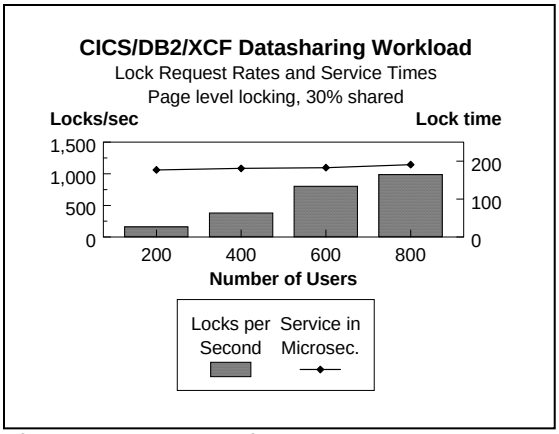

Figure 18 – XES Lock Request Rates

XES lock requests range between 160 and 1,000 per second for page level locking. Compare this with between 370 and 11,000 XES lock requests per second for row level locking. XES lock service times are also better, between 178 and 190 microseconds for page level, and between 168 and 221 microseconds for row level.

Cache activity for page level locking is shown in Figure 19.

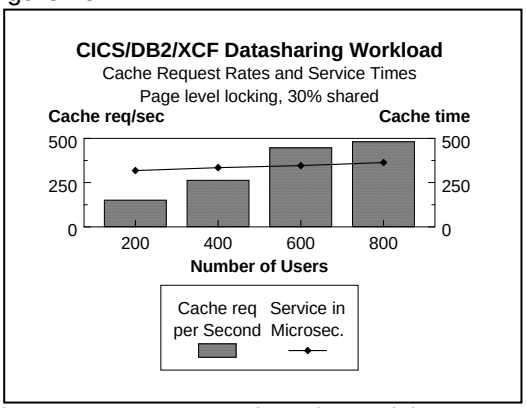

Figure 19 – Page Level Cache Activity

Activity to the global buffer pool via XES Cache requests remained constant at approximately 3 requests per transaction (writing changed pages to the global buffer pool) for row and page level locking, plus another .5 requests per transaction for XI Reads in the page level environment. The XI Reads show up due to the 30% data sharing involved between the two DB2 subsystems. Cache service times range between 316 and 367 microseconds for page level and between 281 and 416 microseconds for row level.

Requests to the IXC_DEFAULT list structure used in CICS/MRO/XCF processing are shown in Figure 20.

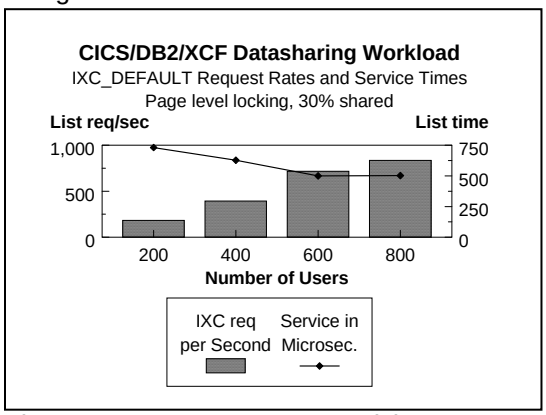

Figure 20 – IXC_DEFAULT Activity

Table 10 following lists some locking numbers provided by the DB2 database statistics record.

#USERS	200	400	600	800
QTXASLAT per transaction				
ROW	.81	.99	1.24	
PAGE	.15	.16	.14	.025
QTGSLPLK per transaction				
ROW	2.92	2.92	3.05	
PAGE	.0047	.0002	.0002	.0002
QTGSKIDS per transaction				
ROW	7.14	4.96	1.77	
PAGE	.3411	.32	.19	.006

Table 10 – Transaction Locking Statistics

QTXASLAT is the number of suspensions due to latch conflicts. Notice that this number is substantially higher in the row-locking environment, and increases with activity. This increase in latch conflicts per transaction may be related to the number of physical locks per transaction, QTGSLPLK. "Page set physical locks are used to track inter-system read-write interest at the page set or partition level to determine if that page set or partition should be GBP-dependent. Page physical locks are used to allow concurrency for one page when used by different data sharing members at the same time."[3] The number of PLOCKS in the row-locking environment stays consistent at about three per transaction, while this number is almost non-existent in the page-locking environment.

The increased locking/unlocking is also due in part to DB2's design for row level locking. "For example, if row locking is used and a row is updated, an X-mode page physical lock is acquired to ensure that no other member touches the page until the update is recorded. The page physical lock protects the page while its structure is being modified. The X-mode page physical lock is released when the page is written to the GBP. This can be before or during the commit of the transaction that caused the page to be modified."[4]

Even though the two DB2 subsystems are not sharing the same physical pages of data in the row-locking environment, DB2's row-locking design causes PLOCK activity to increase as update activity increases. This in turn may cause more requests to be propagated to XES, as reflected in QTGSKIDS. This field is "The number of resources propagated by IRLM to MVS XES asynchronously, including logical and physical locks. This can happen when new inter-DB2 interest occurs on a parent resource or when a request completes after the requester's execution unit was suspended."[5]

8. CONCLUSIONS

The use of Type II indexes opens the door to many of the data-sharing facilities available with DB2 version 4. New statistics are provided, both in RMF and DB2PM, to help determine the benefits and associated resource cots for implementing Type-II indexes. Row-level locking, one of the important areas addressed by the new index type, can help to provide more concurrency in a multi-user environment constrained by concurrency, but can also increase CPU usage.

[3] *DB2 for MVS/ESA Version 4 Data Sharing Performance Topics*, p96, SG24-4611-00

[4] *DB2 for MVS/ESA Version 4 Data Sharing Performance Topics*, p22, SG24-4611-00

[5] *DB2PM Report Reference*, p 79-33, SC26-8986-00

9. REFERENCES

1) DB2 Version 4 Data Sharing: Planning and Administration – SC26-3269-01
2) DB2 Version 4 Release Guide – SC26-3394-01
3) DB2 Version 4 Installation Guide – SC26-3456-00
4) DB2 Version 4 Messages and Codes – SC26-3268-00
5) ITSC System/390 MVS Parallel Sysplex Performance – SG24-4356-01
6) ITSC DB2 for MVS/ESA Version 4 Data Sharing Performance Topics – SG24-4611-00
7) DATABASE 2 Performance Monitor for OS/390 Report Reference, Volume 2 – SC26-89866-00
8) Young, David J. (1991). "Sizing DB2 Applications: Part I." Proceedings of the 17th International Conference for the Management and Performance Evaluation of Computer Systems (pp 1-11). Nashville, Tennessee
9) Young, David J. (1995). "CICSPlex SM/ESA Implementation and Capacity Planning." Proceedings of the 21st International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems (pp 841-852). Nashville, Tennessee