IBM 2007 IMPACT

CICS and VSAM Performance Considerations

Ed Addison

Session 4104A

impact·venture*

Disclaimer

- The workload was run in a controlled environment
  - CICS will put the transaction in a FCIOWAIT each time VSAM issues an I/O
    - This allows the CICS Dispatcher to dispatch other transactions
  - Some workloads have one transaction running on the system
  - Some workloads have multiple tasks running concurrently
    - Response time would vary if other transactions ran between FCIOWAITs
    - CPU per transaction would be more consistent
    - Response time will be greater
    - Throughput will be greater
      –
- VSAM Applications are Shop Dependant. We will not be able to address every possible Application design or access to VSAM
  - Paths
  - Record Keys
  - Record length
  - Batch or Transactional VSAM

Agenda

- Performance Tools used
- NSR
- LSR
- Shared DataTables
- Function Shipping
  - MRO (IRC)
  - LU62 (ISC)
  - MRO/XCF
- RLS

Performance Tools used

Performance Data Sources - CICS

- CICS statistic Records
  - SMF 110 subtype 2 records
  - Information collected on an interval basis and/or end of day
  - Unsolicited Statistics
  - Information is similar to RMF data but CICS based
  - CICS resource based

- CICS monitoring records
  - SMF 110 subtype 1 records
  - CICS task level data - performance and exception
    - DFHFILE
      - Number of file GET, PUT, ADD, Browse and Delete requests issued by the user task
      - FCIOWAIT Time
      - RLS Wait Time
      - CFDT Wait Time

Performance Data Sources – CICS Notes

CICS interval statistics are collected for CICS resource usage at the expiration of each statistics recording interval and written to SMF as type 110 subtype 0002 records. In CICS Transaction Server R2.2, the interval can be specified using the STATINT SIT (System Initialization Table) parameter, STATRCD=ON must also be specified. Otherwise, as is the case with older releases of CICS, the interval is set using the CICS master terminal function CEMT SET STATISTICS, or EXEC CICS SET STATISTICS command.

Consider the interval statistics as CICS region level data, but at a more granular level than RMF data. For example, dataset level statistics vs. the actual DASD activity in RMF.

CICS collects performance data at the task level (activated via the MNPER SIT parm, CEMT SET MONITOR or EXEC CICS SET MONITOR). Three classes of performance monitoring may be selected, performance class data (MNPER), exception class data (MNEXE), and a new transaction resource class data (MNRES), with the addition of PQ63143.

Performance class data is detailed at the transaction level. It provides information such as response time, time spent waiting for a resource or I/O, and CPU time. At least one performance record is written for each transaction at task termination time. For long running tasks, the MNFREQ option can be used to cause periodic records to be written.

Exception class monitoring data provides information about CICS resource shortages at the transaction level. This data can be used to identify system constraints which affect transaction performance. An exception record is written to SMF when the shortage has been resolved. Refer to the CICS Performance Guide for a detailed description of exception records.

CICS Performance Utilities Used

- DFH0STAT
  - QR TCB CPU to Dispatch ratio
  - Transaction rate
  - Reports on many CICS Resources

- CICS Performance Analyzer
  - Performance Post Processor
  - Reports generated for the following:
    - CICS Monitoring Facility data (SMF 110, subtype 1)
    - CICS Statistics data (SMF 110, subtypes 2, 3, 4, 5)
      - 2 – Statistics
      - 3 – Shared Temporary Storage Server Statistics
      - 4 – Coupling Facility Data Table Server Statistics
      - 5 – Named Counter Sequence Number Server Statistics
    - System Logger data (SMF 88 records)
    - DB2 accounting data (SMF 101 records)
    - WebSphere MQSeries accounting data (SMF 116 records)

NSR – Non Shared Resources

Non Shared Resources (NSR)

- NSR allows multiple copies of the same VSAM Control Interval in storage
  - Only one string is allowed to update the Control Interval
  - Other strings are allowed to read copies of the same Control Interval
- Buffer Allocation for NSR
  - Data and Index Buffers are defined on the Resource Definition Online (RDO) FCT Entry
  - Minimum number of Index Buffers is String Number
  - Minimum number of Data Buffers is String Number plus one
    - Extra Data Buffer is reserved for Control Interval Splits
    - Any extra Data Buffers are used for Sequential Processing Read Ahead
- NSR invokes VSAM Read Ahead Processing for Sequential Requests
  - Read Ahead may improve Transaction Response time
  - VSAM will chain together Data Buffers with one I/O
    - Bringing the next sequential Data Buffers into storage decreases I/O Wait Time

NSR Buffering for Concurrent Access

		USER A HAS	
		Exclusive Control	Shared
User B Wants	Exclusive Control	User B gets Exclusive Control Conflict	User B gets second copy of Buffer
	Shared	User B gets second copy of Buffer	User B gets second copy of Buffer

Workload for NSR

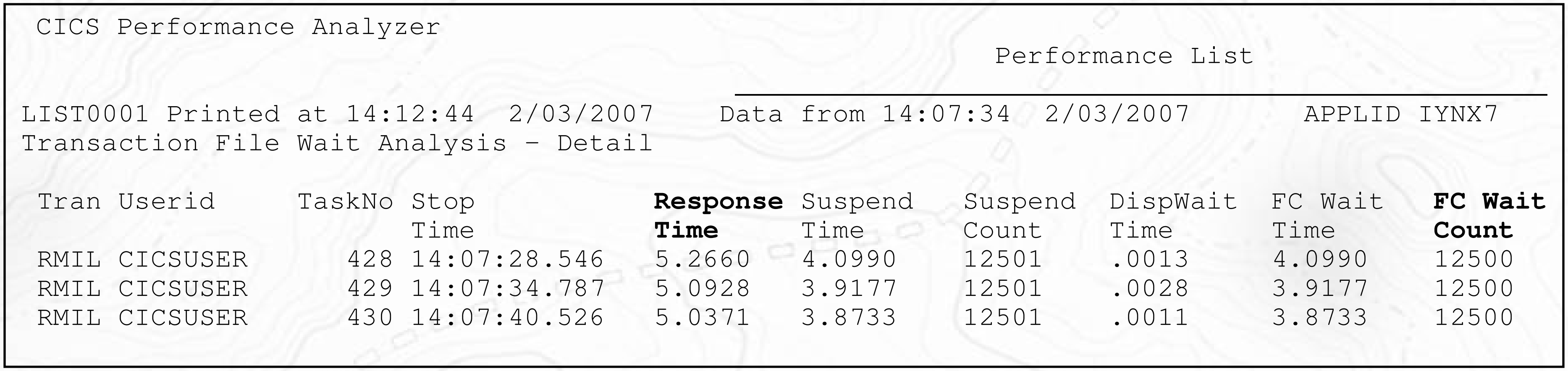
- Dataset Attributes
  - Data Control Interval Size of 4096 Bytes
  - Fixed Record Size of 500 Bytes
    - Eight Records per Control Interval
  - 100,000 Records in the Dataset
    - 12,500 Control Intervals for the Records

- Workload Attributes
  - Read entire dataset from beginning to end
    - 100,000 records
  - Sequential Processing with various Data and Index Buffers
  - Sequential Processing with Concurrent Access
  - Direct Processing

- Performance Tool Used
  - CICS Performance Analyzer

NSR Workload One

- RMIL Transaction that Reads Sequentially the entire 100,000 records
  - EXEC CICS StartBrowse followed by 100,000 EXEC CICS ReadNext
- Default CICS Buffers of 2 Data and 1 Index

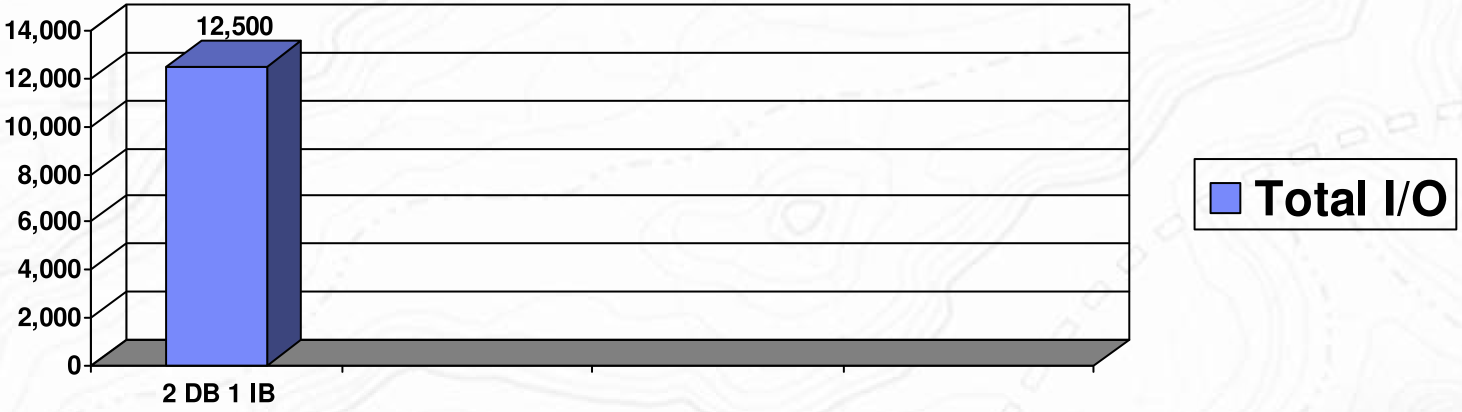


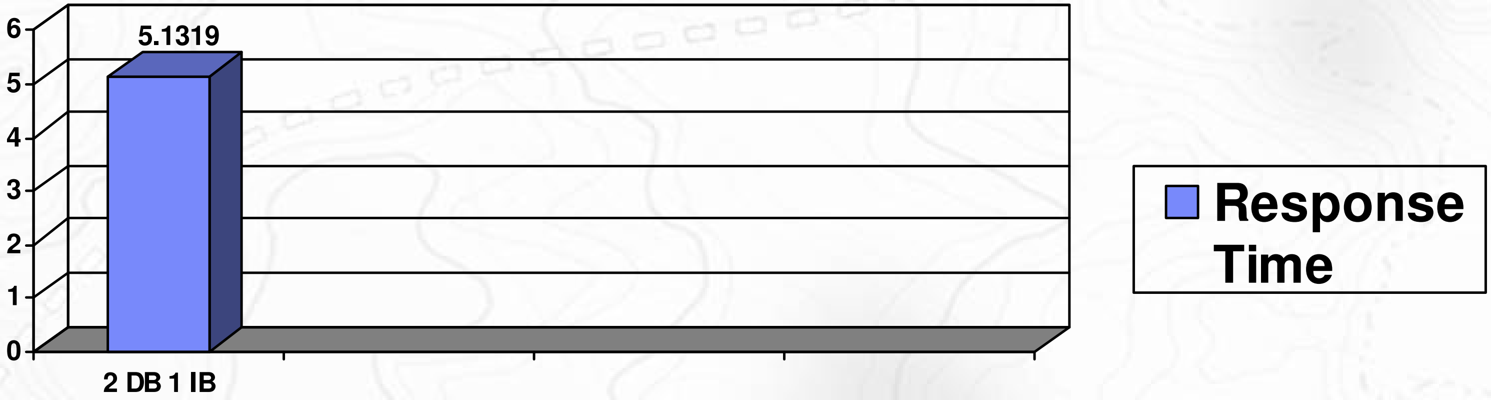
```
 CICS Performance Analyzer
                                                           Performance List
                                                    __________________________________________________
LIST0001 Printed at 14:12:44  2/03/2007    Data from 14:07:34  2/03/2007      APPLID IYNX7
Transaction File Wait Analysis - Detail

 Tran Userid        TaskNo Stop           Response Suspend   Suspend  DispWait  FC Wait   FC Wait
                           Time           Time     Time      Count    Time      Time      Count
 RMIL CICSUSER         428 14:07:28.546   5.2660   4.0990    12501    .0013     4.0990    12500
 RMIL CICSUSER         429 14:07:34.787   5.0928   3.9177    12501    .0028     3.9177    12500
 RMIL CICSUSER         430 14:07:40.526   5.0371   3.8733    12501    .0011     3.8733    12500
```

- With Default Data Buffers there was an I/O Operation for every new CI Read
  - 100,000 records divided by 8 records per CI equals 12,500
- No Read Ahead Processing with Default Data Buffers

NSR Workload Chart – Run One – Single Transaction





NSR Workload Two

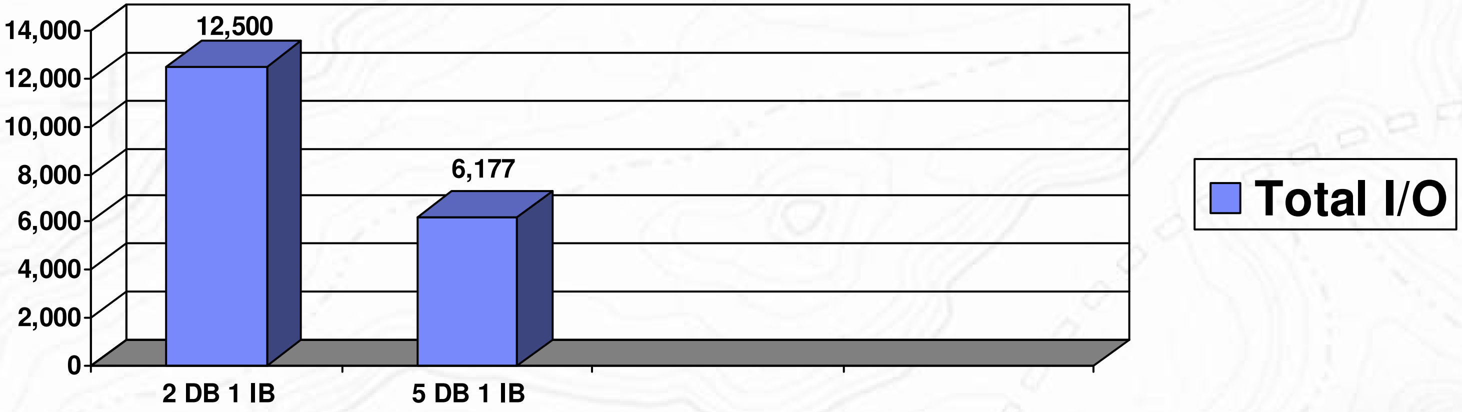
- RMIL Transaction that Reads Sequentially the entire 100,000 records
  - EXEC CICS StartBrowse followed by 100,000 EXEC CICS ReadNext
- Increase Data Buffers to 5 and keep 1 Index Buffer

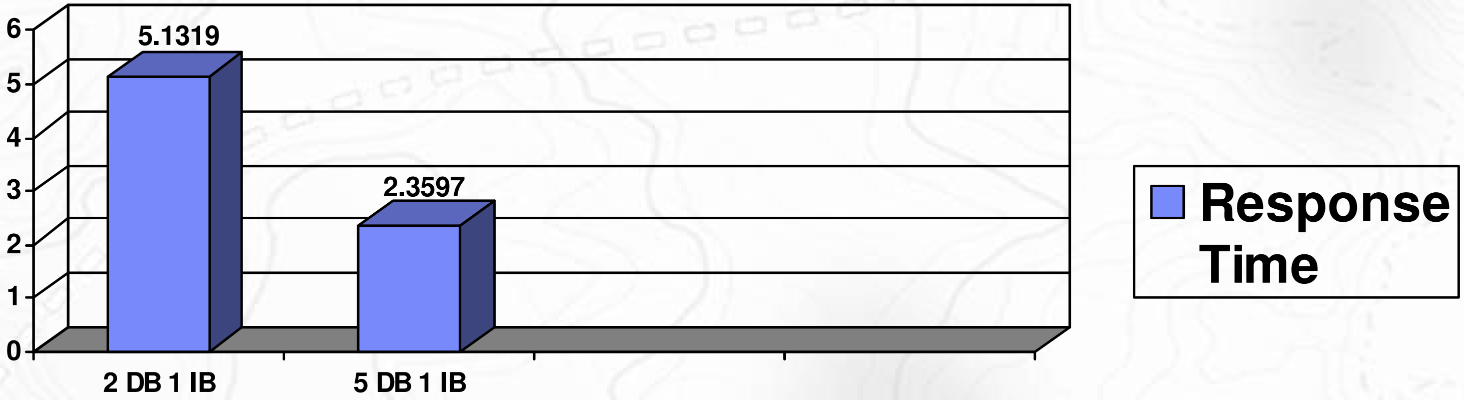
```
 CICS Performance Analyzer
                                                           Performance List
                                                     __________________________________________________
LIST0001 Printed at 16:49:32  2/03/2006    Data from 16:47:40  2/03/2006         APPLID IYNX7
Transaction File Wait Analysis - Detail

 Tran Userid   TaskNo Stop           Response Suspend    Suspend DispWait  FC Wait    FC Wait
                      Time           Time     Time       Count    Time     Time       Count
 RMIL CICSUSER    441 16:47:40.776   2.3708   1.3620     6178     .0008    1.3620     6177
 RMIL CICSUSER    442 16:47:43.597   2.3496   1.3485     6181     .0003    1.3485     6180
 RMIL CICSUSER    443 16:47:46.382   2.3588   1.3542     6179     .0047    1.3542     6178
```

- With Just 5 Data Buffers there was improvement in Response Time and File Control Wait Counts
  - 100,000 records read with 6,177 I/O
- Sequential Read Ahead Processing with just 3 added Data Buffers was successful

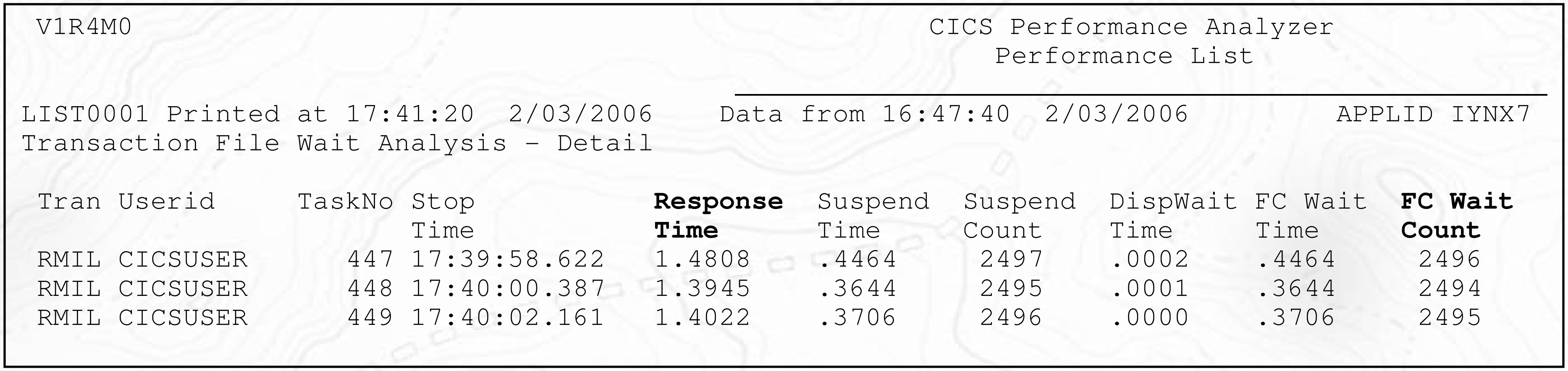
NSR Workload Chart – Run One and Two – Single Transaction





NSR Workload Three

- RMIL Transaction that Reads Sequentially the entire 100,000 records
  - EXEC CICS StartBrowse followed by 100,000 EXEC CICS ReadNext
- Increase Data Buffers to 10 and keep 1 Index Buffer

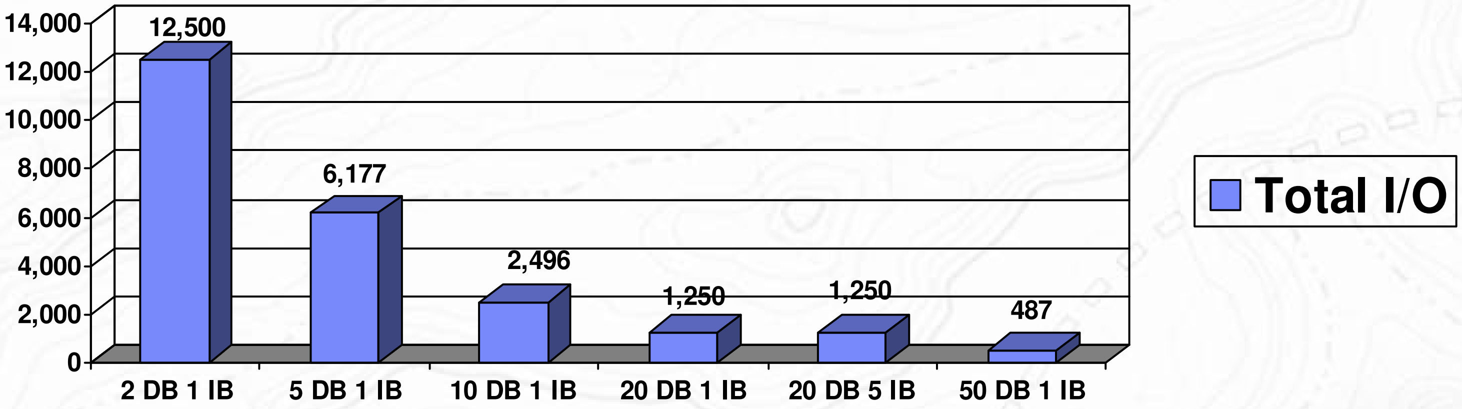


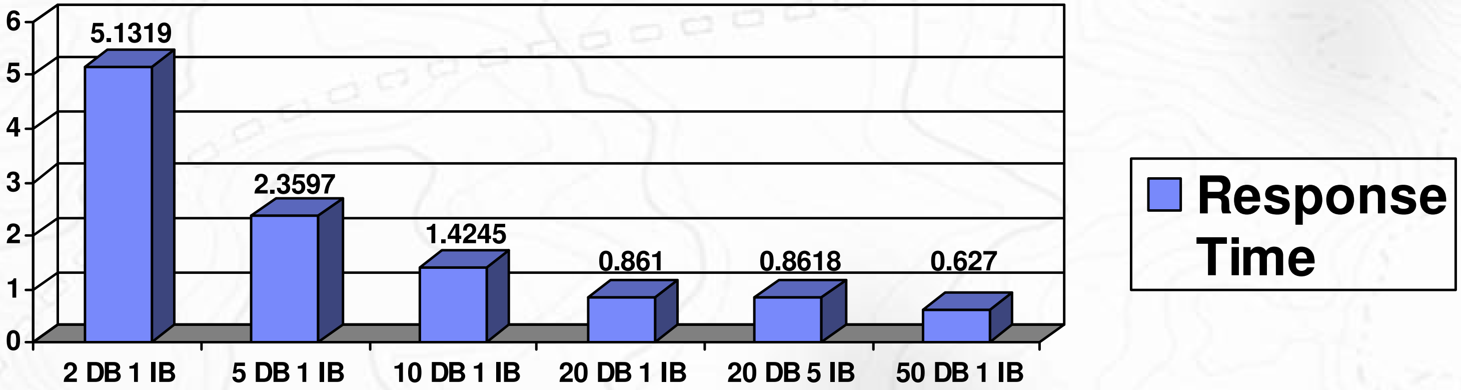
```
 V1R4M0                                                  CICS Performance Analyzer
                                                             Performance List
                                                   ____________________________________________________
LIST0001 Printed at 17:41:20  2/03/2006    Data from 16:47:40  2/03/2006         APPLID IYNX7
Transaction File Wait Analysis - Detail

 Tran Userid      TaskNo Stop          Response Suspend  Suspend  DispWait FC Wait  FC Wait
                         Time          Time     Time     Count    Time     Time     Count
 RMIL CICSUSER       447 17:39:58.622  1.4808   .4464     2497    .0002    .4464     2496
 RMIL CICSUSER       448 17:40:00.387  1.3945   .3644     2495    .0001    .3644     2494
 RMIL CICSUSER       449 17:40:02.161  1.4022   .3706     2496    .0000    .3706     2495
```

- 10 Data Buffers showed another improvement in Response Time and File Control Wait Counts
  - –100,000 records read with 2496 I/O
- Sequential Read Ahead Processing with added Data Buffers still improving

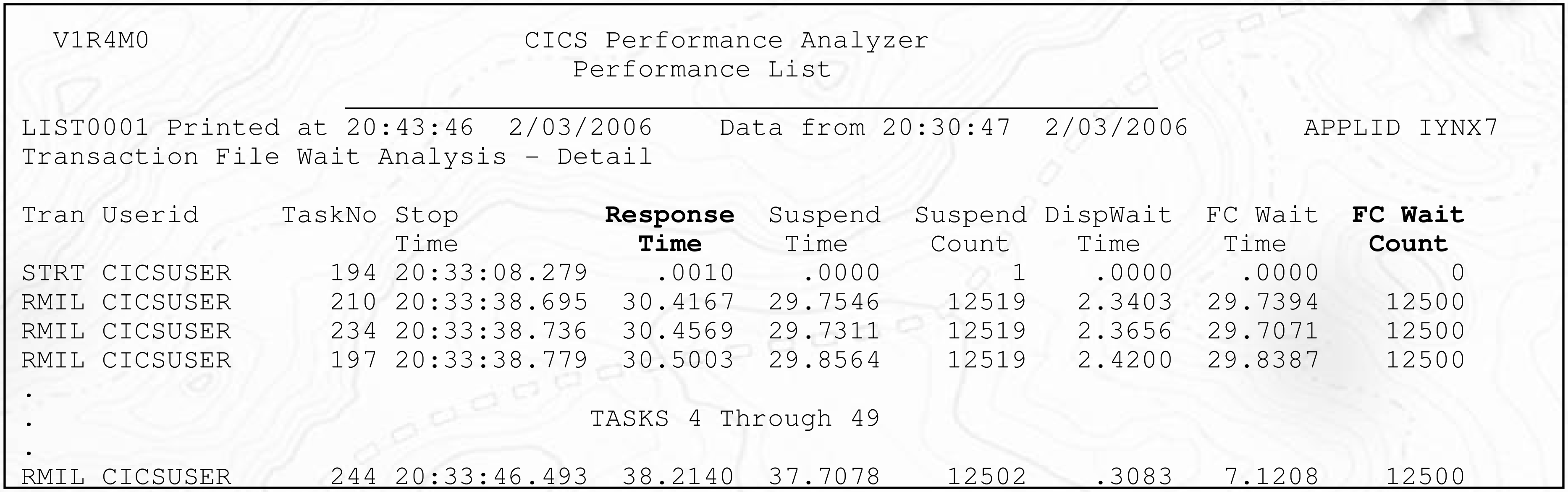
NSR Workload Chart – All Sequential Runs with Single Transaction





NSR Concurrent Workload – Run One

- RMIL Transaction that Reads Sequentially the entire 100,000 records
  - EXEC CICS StartBrowse followed by 100,000 EXEC CICS ReadNext
  - Increased String Number on File to 50 and started 50 concurrent RMIL Transactions
- Default CICS Buffers of 51 Data and 50 Index



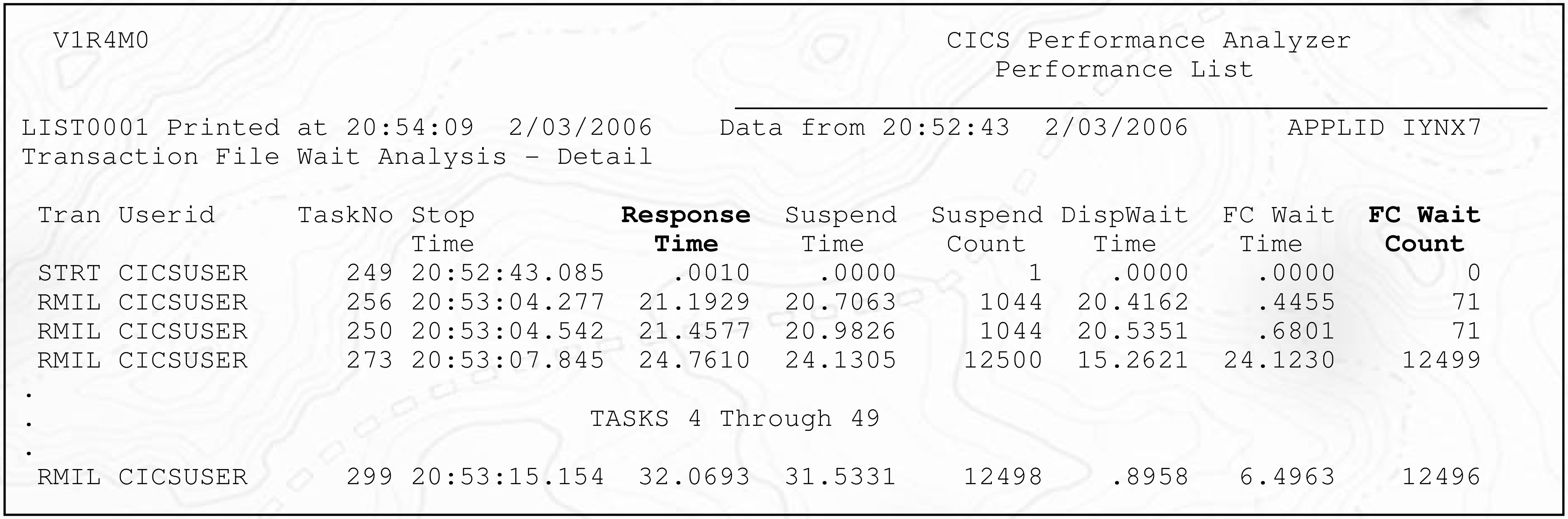
```
  V1R4M0                       CICS Performance Analyzer
                                  Performance List
                    __________________________________________________
LIST0001 Printed at 20:43:46  2/03/2006    Data from 20:30:47  2/03/2006       APPLID IYNX7
Transaction File Wait Analysis - Detail

Tran Userid        TaskNo Stop            Response  Suspend  Suspend DispWait  FC Wait  FC Wait
                          Time              Time      Time    Count    Time     Time     Count
STRT CICSUSER         194 20:33:08.279     .0010     .0000        1    .0000    .0000        0
RMIL CICSUSER         210 20:33:38.695   30.4167   29.7546    12519   2.3403  29.7394    12500
RMIL CICSUSER         234 20:33:38.736   30.4569   29.7311    12519   2.3656  29.7071    12500
RMIL CICSUSER         197 20:33:38.779   30.5003   29.8564    12519   2.4200  29.8387    12500
.
.                                  TASKS 4 Through 49
.
RMIL CICSUSER         244 20:33:46.493   38.2140   37.7078    12502    .3083   7.1208    12500
```

- With Default Data Buffers there was an I/O Operation for every new CI Read
  - 100,000 records divided by 8 records per CI equals 12,500
  - Completed 50 transactions in 38.2 seconds
- No Read Ahead Processing with Default Data Buffers

NSR Concurrent Workload – Run Two

- RMIL Transaction that Reads Sequentially the entire 100,000 records
  - EXEC CICS StartBrowse followed by 100,000 EXEC CICS ReadNext
  - Increased String Number on File to 50 and started 50 concurrent RMIL Transactions
- Increase Data Buffers of 500 Data and 50 Index



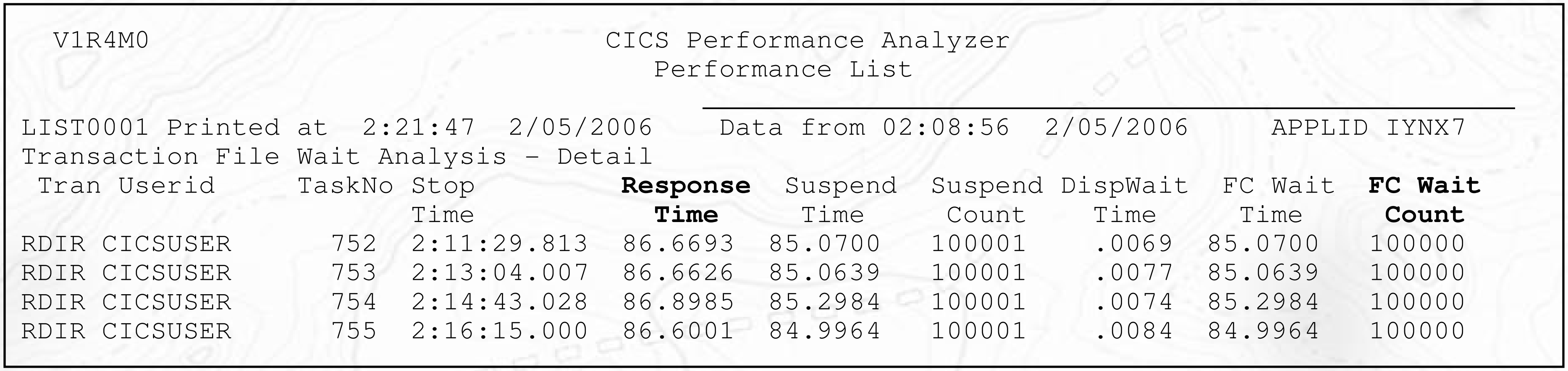
```
  V1R4M0                                                 CICS Performance Analyzer
                                                            Performance List
                                                     ___________________________________________________
LIST0001 Printed at 20:54:09  2/03/2006    Data from 20:52:43  2/03/2006      APPLID IYNX7
Transaction File Wait Analysis - Detail

 Tran Userid      TaskNo Stop          Response  Suspend  Suspend DispWait  FC Wait  FC Wait
                         Time              Time     Time    Count     Time     Time    Count
 STRT CICSUSER       249 20:52:43.085     .0010    .0000        1    .0000    .0000        0
 RMIL CICSUSER       256 20:53:04.277   21.1929  20.7063     1044  20.4162    .4455       71
 RMIL CICSUSER       250 20:53:04.542   21.4577  20.9826     1044  20.5351    .6801       71
 RMIL CICSUSER       273 20:53:07.845   24.7610  24.1305    12500  15.2621  24.1230    12499
.
.                                     TASKS 4 Through 49
.
 RMIL CICSUSER       299 20:53:15.154   32.0693  31.5331    12498    .8958   6.4963    12496
```

- With 500 Data Buffers there was Read Ahead processing for some Transactions
  - First few and some middle RMIL Transactions benefited from Read Ahead
  - Completed 50 transactions in 32.06 seconds for throughput of .6412 per transaction

NSR Direct Processing Workload

- RDIR Transaction that Reads directly the entire 100,000 records
  - Read the entire dataset with 100,000 EXEC CICS READ commands
  - String Number on File set to 5 with 20 Data and 5 Index Buffers
  - Transactions submitted non-concurrent



```
  V1R4M0                                        CICS Performance Analyzer
                                                   Performance List
                                                ___________________________________________________
LIST0001 Printed at  2:21:47  2/05/2006    Data from 02:08:56  2/05/2006     APPLID IYNX7
Transaction File Wait Analysis - Detail
 Tran Userid      TaskNo Stop          Response  Suspend  Suspend DispWait  FC Wait  FC Wait
                         Time            Time      Time     Count    Time     Time     Count
RDIR CICSUSER      752  2:11:29.813  86.6693  85.0700   100001    .0069  85.0700   100000
RDIR CICSUSER      753  2:13:04.007  86.6626  85.0639   100001    .0077  85.0639   100000
RDIR CICSUSER      754  2:14:43.028  86.8985  85.2984   100001    .0074  85.2984   100000
RDIR CICSUSER      755  2:16:15.000  86.6001  84.9964   100001    .0084  84.9964   100000
```

- With Direct Access and 20 Data Buffers there is no Read Ahead processing
  - Same exact workload changed to Direct Processing
  - Response time went from .8618 seconds to 86 seconds
  - FCIOWAITs went from 1,250 to 100,000
- CICS has to reestablish the string sent to VSAM on every request
  - With NSR and direct there is no concept of a Buffer already in storage

NSR Sequential Processing with Non-Concurrent and Concurrent Transactions Observations

- Non-Concurrent Transaction
  - Physical I/O to DASD and Response Time decreased as Data Buffers increased
    - Due to VSAM invoking Read Ahead Logic for Sequential NSR access
    - Requires tuning to find the exact number of Data Buffers that help Read Ahead Processing without taking up storage resources
  - Increase of Index Buffers has no effect on Response Time
    - During Sequential Processing VSAM gets to the next Index CI by using the horizontal pointers in sequence set records rather than the vertical pointers in the index set
  - While the Transaction was in an FCIOWAIT there was nothing else for CICS to dispatch
    - The wait for I/O to complete is a long time for the CICS Dispatcher to be idle

- Concurrent Transactions
  - Physical I/O to DASD did not have as dramatic of an effect when 50 Transactions competed for the 500 Data Buffers
  - Other Transactions could be dispatched while Transactions are in an FCIOWAIT
    - Response time per Transaction was up from 2.3597 (5 DB 1 IB) to 21 through 32 second range (500 DB 50 IB 50 Strings)
    - 50 Transactions did complete in 32.06 seconds wall clock time
      - Compare to roughly two minutes to complete non-concurrent
      - While one task was in an FCIOWAIT other tasks were able to run

NSR Summary

- Sequential NSR processing is a good match for workload that consists of StartBrowse / ReadNext activity
  - VSAM will also chain I/O for MassInsert activity (Sequential Writes)
- Read Ahead processing may decrease as concurrent users increase
- NSR is not a good match for Direct Processing
  - Response time of reading 100,000 records with 20 Data Buffers and 5 Index Buffers went from .8618 seconds to over 86 seconds
  - Physical I/O went from 1,250 to 100,000
    - NSR does not have a concept of Data Buffer already in storage for Direct Processing
- Allows specific tuning of a particular dataset
- NSR cannot be ran with Transaction Isolation active
  - See Information APAR II09208
- Read Integrity issues are more prevalent with NSR
  - An Update request and Read request would each have their own copy of a Data Buffer
  - Read request would not have knowledge of the updated Data Buffer

LSR – Local Shared Resources

Local Shared Resources (LSR)

- LSR provides more efficient use of storage because buffers and strings are shared among Transactions
  - Many Read / Browse Transactions can share records in Data and Index Buffers
    - Transactions wanting to update the Buffers will be put into an FCXCWAIT
  - Only one Transaction can update the Data and Index Buffers
    - Transactions wanting to read a record in the Buffer will be put into an FCXCWAIT
- Buffer Allocation for LSR
  - Buffers are defined using Resource Definition Online (RDO) entry LSRPOOL
    - Buffers are either User defined or will be determined by CICS
- LSR has concept of Buffer already in storage
  - Reads and Browses will benefit from Lookaside processing
    - VSAM will not issue an I/O if the Buffer is already in storage
      - Note: Shareoption 4 will force an I/O for Read requests
  - VSAM will use a Least Recently Used (LRU) algorithm to bring new buffers into the Pool

LSR Buffering for Concurrent Access

		USER A HAS	
		Exclusive Control	Shared
User B Wants	Exclusive Control	User B gets Exclusive Control Conflict	User B is queued until User A releases Buffer Or, User B receives FCXCWAIT Note(1)
	Shared	User B gets Exclusive Control Conflict	User B shares same buffer with User A

Note(1): CICS always sets ACBNLW (No LSR Wait) in the ACB control block. For CICS, User B will receive an FCXCWAIT

Workload for LSR

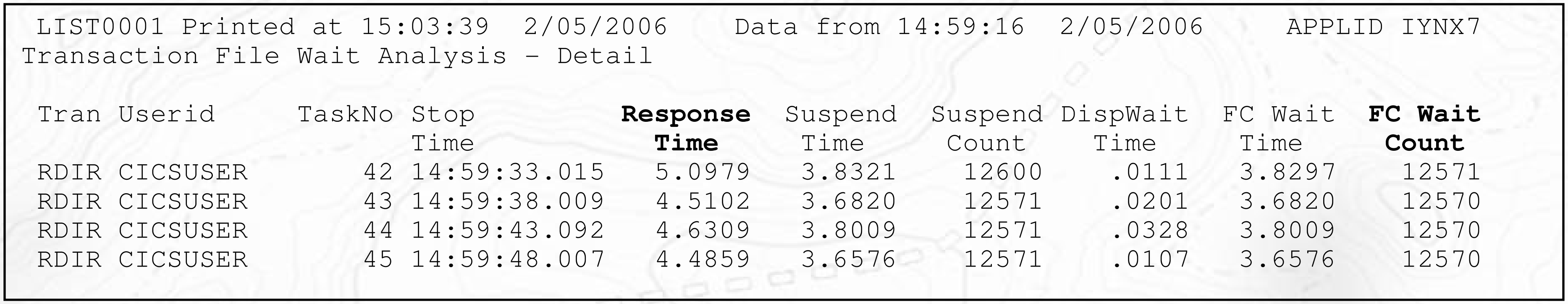
- Dataset Attributes
  - Data Control Interval Size of 4096 Bytes
  - Fixed Record Size of 500 Bytes
    - Eight Records per Control Interval
  - 100,000 Records in the Dataset
    - 12,500 Control Intervals for the Records

- Workload Attributes
  - Read entire dataset from beginning to end
    - 100,000 records
  - Sequential Processing with various Data and Index Buffers
  - Sequential Processing with Concurrent Access
  - Direct Processing

- Performance Tool Used
  - CICS Performance Analyzer
  - CICS STAT Transaction

LSR Workload One

- Same RDIR Transaction that Reads the entire 100,000 records
  - 100,000 EXEC CICS READ commands for different records
- Default CICS Built LSRPOOL (3 Data and 4 Index Buffers)

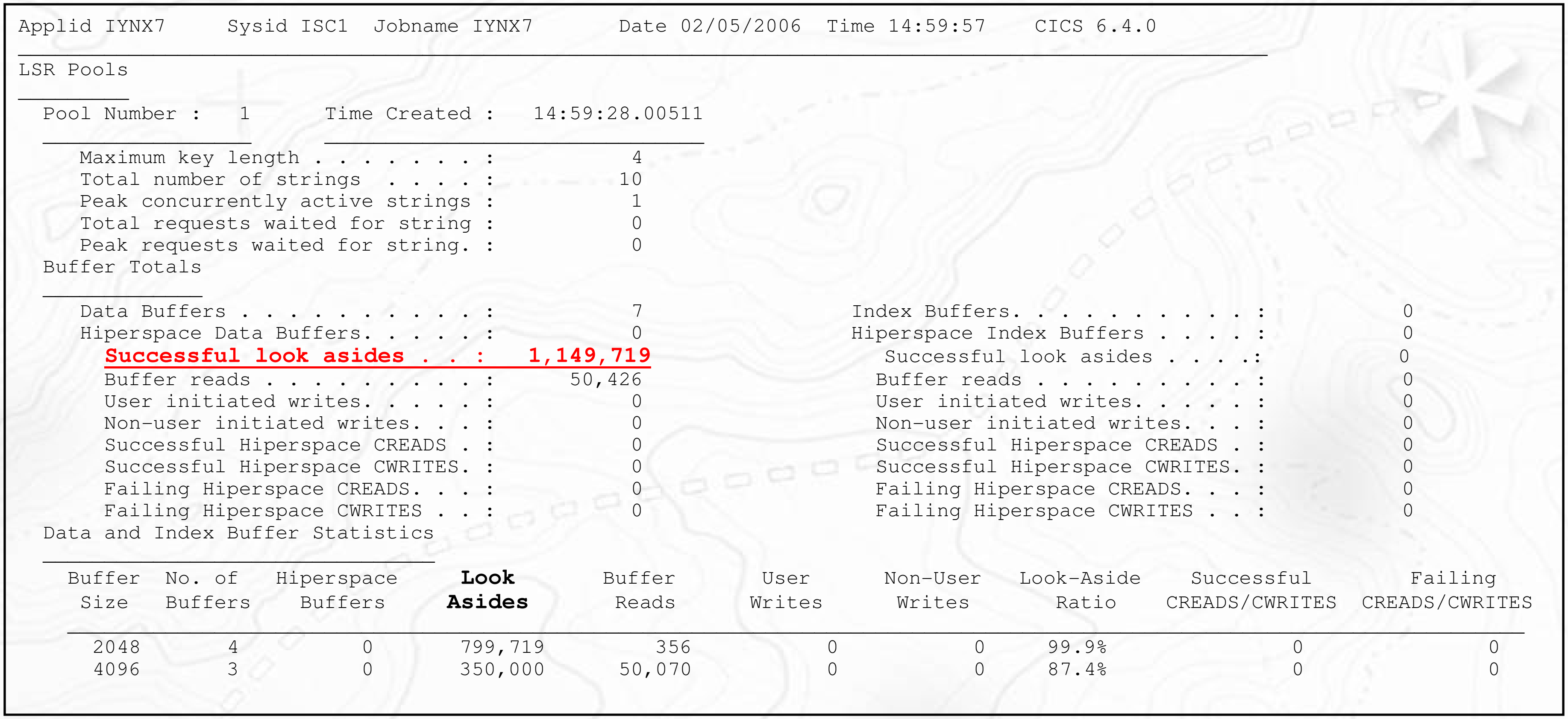


```
 LIST0001 Printed at 15:03:39  2/05/2006    Data from 14:59:16  2/05/2006     APPLID IYNX7
Transaction File Wait Analysis - Detail

 Tran Userid       TaskNo Stop          Response  Suspend  Suspend DispWait  FC Wait  FC Wait
                          Time            Time     Time     Count    Time     Time     Count
 RDIR CICSUSER         42 14:59:33.015   5.0979   3.8321    12600    .0111   3.8297    12571
 RDIR CICSUSER         43 14:59:38.009   4.5102   3.6820    12571    .0201   3.6820    12570
 RDIR CICSUSER         44 14:59:43.092   4.6309   3.8009    12571    .0328   3.8009    12570
 RDIR CICSUSER         45 14:59:48.007   4.4859   3.6576    12571    .0107   3.6576    12570
```

- Same workload in NSR had a Response Time of over 86 Seconds
  - Compare to less than 5 seconds in LSR
- Same workload in NSR had 100,000 I/Os per Transaction
  - VSAM invoked Lookaside Processing since records requested were already in Buffers
    - Data Buffer Lookaside was 87.4%
    - Index Buffer Lookaside was 99.9%

DFH0STAT Output – LSRPOOL for LSR Workload One

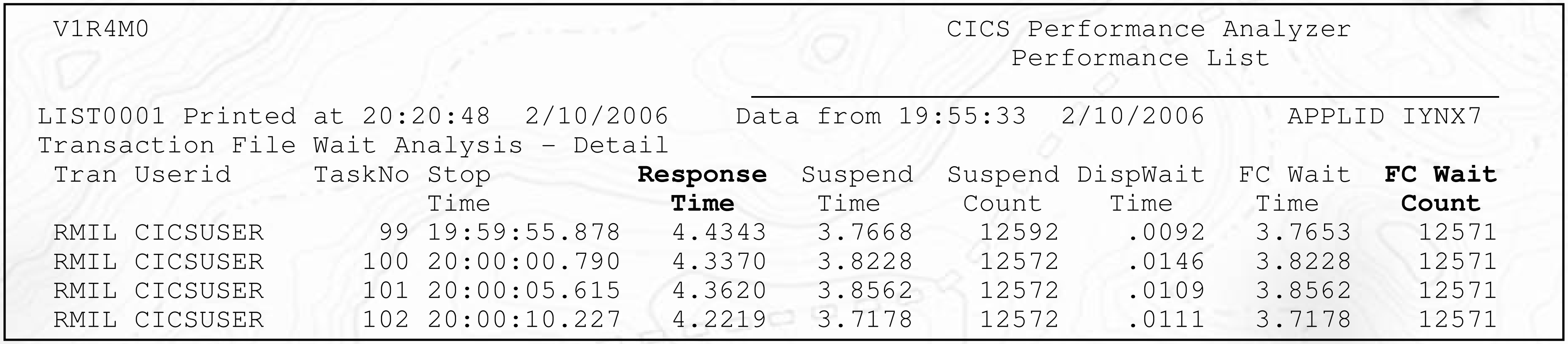


```
Applid IYNX7     Sysid ISC1   Jobname IYNX7       Date 02/05/2006   Time 14:59:57    CICS 6.4.0
____________________________________________________________________________________________________
LSR Pools
_________
  Pool Number :   1       Time Created :   14:59:28.00511
  _________________       _______________________________
     Maximum key length . . . . . . . :             4
     Total number of strings  . . . . :            10
     Peak concurrently active strings :             1
     Total requests waited for string :             0
     Peak requests waited for string. :             0
  Buffer Totals
  _____________
     Data Buffers . . . . . . . . . . :             7                  Index Buffers. . . . . . . . . . :            0
     Hiperspace Data Buffers. . . . . :             0                  Hiperspace Index Buffers . . . . :            0
       Successful look asides . . :   1,149,719                          Successful look asides . . . .:            0
       Buffer reads . . . . . . . . . :        50,426                    Buffer reads . . . . . . . . . :            0
       User initiated writes. . . . . :             0                    User initiated writes. . . . . :            0
       Non-user initiated writes. . . :             0                    Non-user initiated writes. . . :            0
       Successful Hiperspace CREADS . :             0                    Successful Hiperspace CREADS . :            0
       Successful Hiperspace CWRITES. :             0                    Successful Hiperspace CWRITES. :            0
       Failing Hiperspace CREADS. . . :             0                    Failing Hiperspace CREADS. . . :            0
       Failing Hiperspace CWRITES . . :             0                    Failing Hiperspace CWRITES . . :            0
  Data and Index Buffer Statistics
  ________________________________
    Buffer  No. of   Hiperspace     Look       Buffer      User     Non-User   Look-Aside    Successful       Failing
     Size   Buffers    Buffers     Asides       Reads     Writes     Writes      Ratio     CREADS/CWRITES  CREADS/CWRITES
    _______________________________________________________________________________________________________________________
     2048       4          0      799,719         356          0          0     99.9%              0               0
     4096       3          0      350,000      50,070          0          0     87.4%              0               0
```

Note: With the minimum amount of CICS determined buffers there were 1,149,719 Lookaside hits for the Reads.
Index Lookaside is great, but data Lookaside could still be improved with added buffers

LSR Workload Two

- RMIL Transaction that Sequentially Reads the entire 100,000 records
  - EXEC CICS StartBrowse followed by 100,000 ReadNext commands
- User Built LSRPOOL (50 Data and 10 Index Buffers)



```
  V1R4M0                                                    CICS Performance Analyzer
                                                                Performance List
                                                   ______________________________________________
 LIST0001 Printed at 20:20:48  2/10/2006    Data from 19:55:33  2/10/2006      APPLID IYNX7
 Transaction File Wait Analysis - Detail
  Tran Userid      TaskNo Stop           Response  Suspend  Suspend DispWait  FC Wait  FC Wait
                          Time             Time      Time     Count     Time     Time    Count
  RMIL CICSUSER        99 19:59:55.878   4.4343   3.7668    12592    .0092   3.7653    12571
  RMIL CICSUSER       100 20:00:00.790   4.3370   3.8228    12572    .0146   3.8228    12571
  RMIL CICSUSER       101 20:00:05.615   4.3620   3.8562    12572    .0109   3.8562    12571
  RMIL CICSUSER       102 20:00:10.227   4.2219   3.7178    12572    .0111   3.7178    12571
```

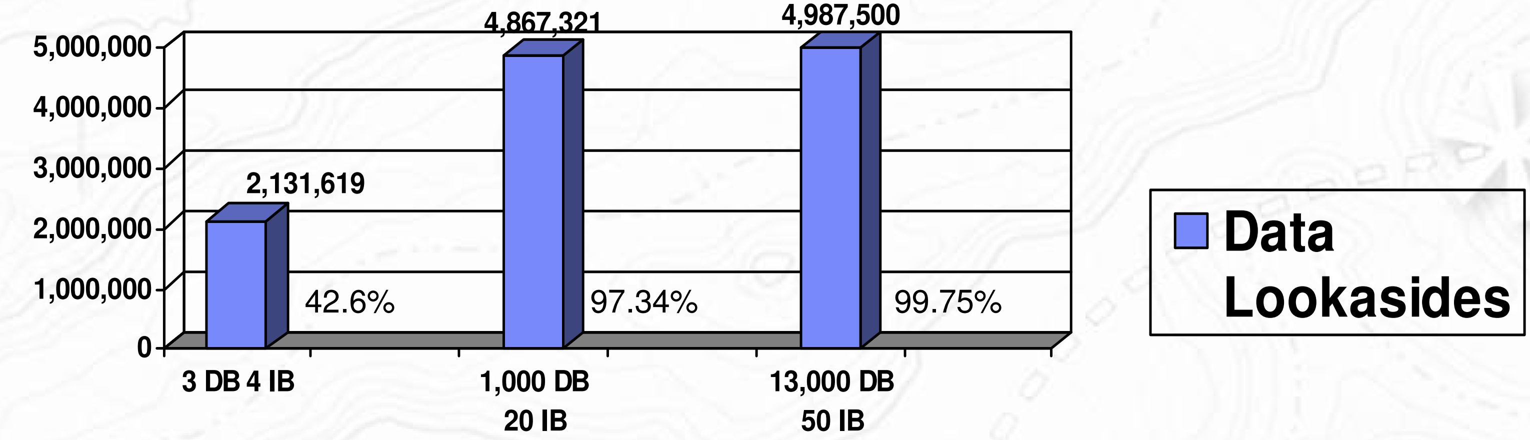
- Same workload in NSR had a Response .627 Seconds
  - Compare to over 4 seconds in LSR
- Same workload in NSR had 487 I/Os per Transaction
  - VSAM invoked Lookaside Processing since records requested were already in Buffers
    - Data Buffer Lookaside was 0%
    - Index Buffer Lookaside was 99.4%

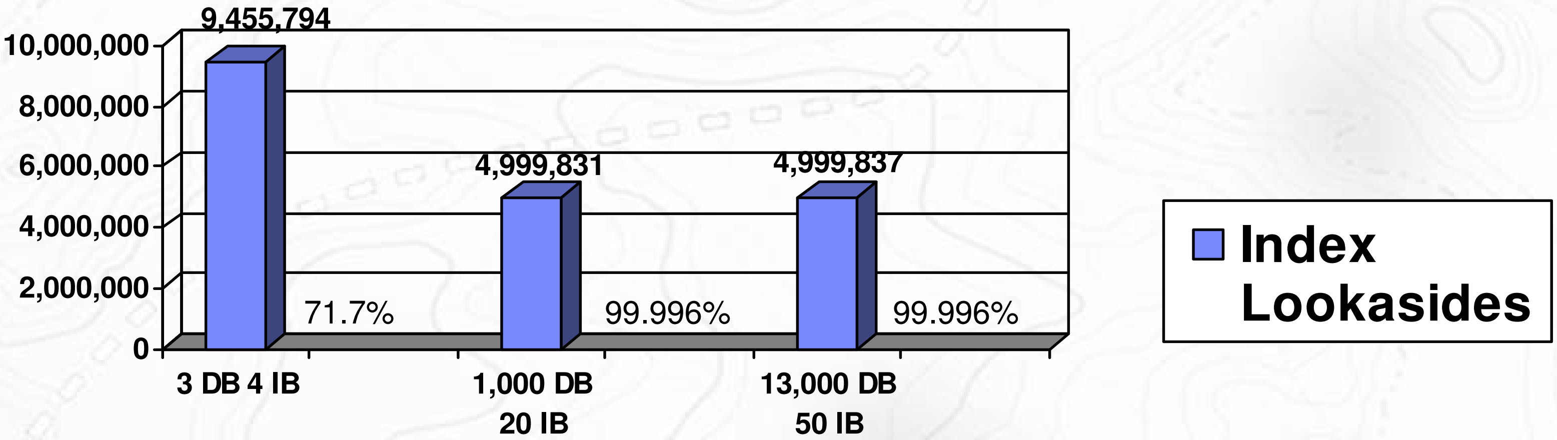
DFH0STAT Output – LSRPOOL for LSR Workload Two

```
LSR Pools
_________
  Pool Number :    4       Time Created :    19:59:51.50066
  _________________       _______________________________
     Maximum key length . . . . . . . . :            25
     Total number of strings  . . . . . :           100
     Peak concurrently active strings :               1
     Total requests waited for string :               0
     Peak requests waited for string. :               0
  Buffer Totals
  _____________
     Data Buffers . . . . . . . . . . . :            50                        Index Buffers. . . . . . . . . . . :            20
     Hiperspace Data Buffers. . . . . . :             0                        Hiperspace Index Buffers . . . . . :             0
       Successful look asides . . . . :              0                           Successful look asides . . . . :        49,996
       Buffer reads . . . . . . . . . :         50,001                           Buffer reads . . . . . . . . . :           289
       User initiated writes. . . . . :              0                           User initiated writes. . . . . :             0
       Non-user initiated writes. . . :              0                           Non-user initiated writes. . . :             0
       Successful Hiperspace CREADS . :              0                           Successful Hiperspace CREADS . :             0
       Successful Hiperspace CWRITES. :              0                           Successful Hiperspace CWRITES. :             0
       Failing Hiperspace CREADS. . . :              0                           Failing Hiperspace CREADS. . . :             0
       Failing Hiperspace CWRITES . . :              0                           Failing Hiperspace CWRITES . . :             0
 Data Buffer Statistics
 ______________________
   Buffer   No. of    Hiperspace      Look        Buffer      User      Non-User    Look-Aside     Successful        Failing
    Size    Buffers     Buffers      Asides        Reads     Writes      Writes        Ratio      CREADS/CWRITES   CREADS/CWRITES
   ____________________________________________________________________________________________________________________________
     4096      50          0            0         50,001        0           0        0.0%              0                0
 Index Buffer Statistics
 _______________________
   Buffer   No. of    Hiperspace      Look        Buffer      User      Non-User    Look-Aside     Successful        Failing
    Size    Buffers     Buffers      Asides        Reads     Writes      Writes        Ratio      CWRITES/CREADS   CREADS/CWRITES
   ____________________________________________________________________________________________________________________________
     2048      20          0         49,996          289        0           0        99.4%              0                0
```

Note: VSAM does not count a sequential Read that is in a Data Buffer as a Lookaside hit. This is due to the fact that the string was positioned in the Data Buffer and there was no need to find the record. Also note that there is 12,500 I/O per run (50,001 / 4 runs). This will never decrease in LSR for Sequential Processing for this single transaction.

LSR Workload Chart – 50 Concurrent RDIR Transactions





Note: The reason there were more Lookasides when there was 4 Index Buffers is due to all 50 tasks competing for the buffers. There was a lot of buffer steals causing extra DASD I/O. The Lookaside ratio for the Index was 71.7% with 3,715,192 Reads to DASD. The Lookaside ratio for the Data was 42.6% with 2,868,381 Reads to DASD.

LSR Summary

- LSR processing is a good match for workload that consists of Direct Reads
  - Response time for 100,000 Direct Reads in NSR was over 86 seconds compared to under 5 seconds in LSR
- Lookaside processing may increase as concurrent users increase
- LSR is not a great match for Sequential Processing
  - Response time of reading 100,000 records sequentially with a well tuned LSRPOOL was over four seconds while Physical I/O was 12,500
    - Same workload in NSR had a Response .627 Seconds
    - Same workload in NSR had 487 I/Os
  - VSAM does not chain together I/Os for LSR
- LSR can be ran with Transaction Isolation active
- Read Integrity issues are less prevalent with LSR
  - There is only one copy of a VSAM Control Interval in storage
  - Read requests will wait until an update request is finished before gaining access to the records in the Control Interval

LSR Summary

- LSR provides the following
  - More efficient use of virtual storage because buffers and strings are shared
  - Better performance because of buffer lookaside, which can reduce I/O operations
  - Self-tuning because more buffers are allocated to busy files and frequently referenced index control intervals are kept in the LSRPOOL
    - If VSAM needs to steal a buffer it will choose the Least Recently Used buffer
  - Use of synchronous file requests and a UPAD exit. CA and CI splits for LSR files do not cause either the subtask or main task to wait
    - VSAM takes the UPAD exit while waiting for physical I/O, and processing continues for other CICS work during the CA/CI split
- LSR is susceptible to Exclusive Control Conflicts
  - An update request will receive an FCXCWAIT if there is a Browse or Read active in the Buffer it wants to update
  - A Read or Browse request will receive an FCXCWAIT if there is an update active in the buffer it wants to access

Shared Data Tables

Shared Data Tables

- Shared Data Tables provides
  - Usage of MVS(TM) cross-memory services instead of CICS function shipping to share a file of data between two or more CICS regions in the same MVS image
  - Access of records are from memory instead of from DASD
  - Very large reductions in path length can be achieved for remote accesses because function shipping is avoided for most read and browse requests
  - Cross-memory services are used, so the requests are processed by the AOR, thus freeing the FOR to process other requests
  - Any number of files referring to the same source data set that are open at the same time can retrieve data from the one CICS-maintained data table
  - Increased security of data is provided because the record information in Shared Data Tables is stored outside the CICS region and is not included in CICS system dumps (either formatted or unformatted)
  - User Exit XDTRD which allows you to skip over a range of records while loading the data table

Shared Data Tables

- CICS Maintained Data Table
  - Updates are reflected in both the Data Table and the VSAM Source KSDS
    - Full Recovery aspects of the Source KSDS are maintained
    - Source KSDS cannot be accessed in RLS mode
  - No Application Program changes are needed
- User Maintained Data Table
  - Updates are only reflected in the Data Table
    - VSAM Source KSDS is not updated
    - Recovery is only supported after a transaction failure, not a system failure
  - Some File Control requests are not supported, so Application Program changes may be needed
    - Reference CICS Shared Data Table Guide Section 5.2 Application programming for a user-maintained data table
  - Source KSDS can be accessed in RLS mode

Workload for Shared Data Table

- Dataset Attributes
  - Data Control Interval Size of 4096 Bytes
  - Fixed Record Size of 500 Bytes
    - Eight Records per Control Interval
  - 100,000 Records in the Dataset
    - 12,500 Control Intervals for the Records

- Workload Attributes
  - Read entire dataset from beginning to end
    - 100,000 records
  - Sequential Processing
  - Sequential Processing with Concurrent Access
  - Direct Processing
  - Direct Processing with Concurrent Access

- Performance Tool Used
  - CICS Performance Analyzer

Shared Data Table Workload One

- Same RDIR Transaction that Reads the entire 100,000 records
  - 100,000 EXEC CICS READ commands for different records

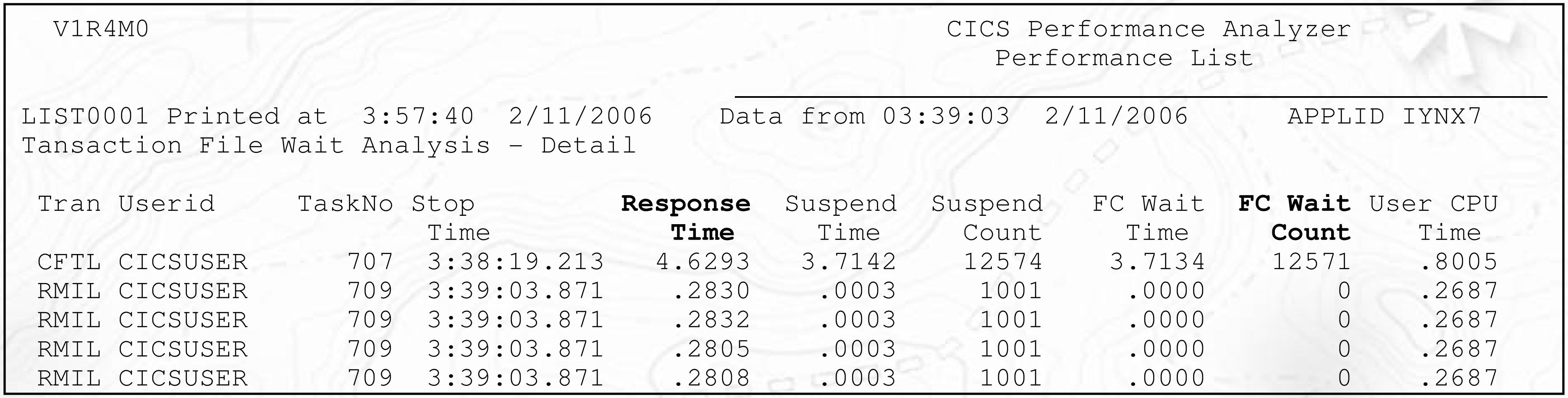
```
 Tran Userid      TaskNo Stop         Response  Suspend  Suspend  FC Wait  FC Wait  User CPU
                           Time           Time     Time    Count     Time    Count     Time
CFTL CICSUSER       711  3:39:32.616   4.5211   3.6776    12572   3.6774    12571     .7728
RDIR CICSUSER       712  3:39:36.514    .2826    .0002     1000    .0000        0     .2679
RDIR CICSUSER       713  3:39:41.426    .2865    .0003     1000    .0000        0     .2676
RDIR CICSUSER       714  3:39:42.561    .2817    .0001     1000    .0000        0     .2647
RDIR CICSUSER       715  3:39:43.832    .2804    .0003     1000    .0000        0     .2617
```

- Same workload in NSR had a Response Time of over 86 Seconds
- Same workload in LSR had a Response time of over 4.5 seconds
  - Compare to .28 seconds using Shared Data Tables
- Same workload in NSR had 100,000 I/Os per Transaction
- Same Workload in LSR had I/Os even with 13,000 Data Buffers

NOTE: CICS Transaction CFTL is used to load the records from the Source KSDS to the DataSpace

Shared Data Tables Workload Two

- RMIL Transaction that Sequentially Reads the entire 100,000 records
  - EXEC CICS StartBrowse followed by 100,000 ReadNext commands



```
  V1R4M0                                                    CICS Performance Analyzer
                                                               Performance List
                                                    ______________________________________________________
LIST0001 Printed at  3:57:40  2/11/2006    Data from 03:39:03  2/11/2006      APPLID IYNX7
Tansaction File Wait Analysis - Detail

 Tran Userid      TaskNo Stop         Response  Suspend  Suspend   FC Wait  FC Wait User CPU
                          Time            Time     Time    Count      Time    Count     Time
 CFTL CICSUSER       707  3:38:19.213   4.6293   3.7142    12574    3.7134    12571    .8005
 RMIL CICSUSER       709  3:39:03.871    .2830    .0003     1001     .0000        0    .2687
 RMIL CICSUSER       709  3:39:03.871    .2832    .0003     1001     .0000        0    .2687
 RMIL CICSUSER       709  3:39:03.871    .2805    .0003     1001     .0000        0    .2687
 RMIL CICSUSER       709  3:39:03.871    .2808    .0003     1001     .0000        0    .2687
```

- Same workload in NSR had a Response time of .627 Seconds
- Same workload in LSR had a response time of over 4 seconds
  - Compare to .28 Seconds using Shared Data Tables
- Same workload in NSR had 487 I/Os per Transaction
- Same workload in LSR had 12,500 I/Os per Transaction
  - Compare to no I/O for Shared Data Tables

Function Shipping

Function Shipping

- CICS Function Shipping Provides
  - Application program access to a resource owned by another CICS system
    - Both read and write access are permitted
    - Facilities for exclusive control and recovery and restart are provided
- The remote resource can be
  - A file
  - A DL/I database
  - A transient-data queue
  - A temporary-storage queue
  - A Transaction using EXEC CICS Start
  - A Program using Dynamic Program Link
  -
- Application programs that access remote resources can be designed and coded as if the resources were owned by the system in which the transaction is to run
  - During execution, CICS Function Ships the request to the appropriate system

Function Shipping – Multi Region Operation (MRO)

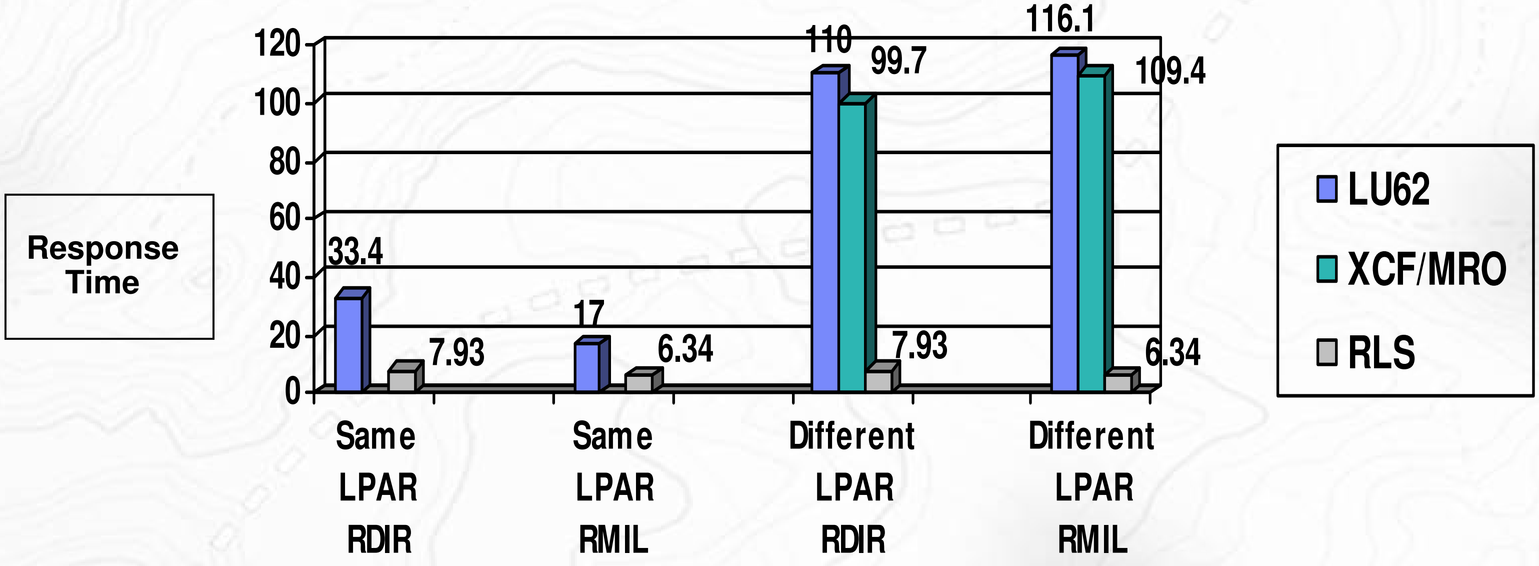
- For CICS-to-CICS communication, CICS provides an **inter-region communication** facility that is independent of SNA access methods called **multi-region operation** (MRO). MRO can be used between CICS systems that reside:
  - In the same host operating system
  - In the same MVS systems complex (**sysplex**) using XCF/MRO
- CICS Transaction Server for z/OS can use MRO to communicate with:
  - Other CICS Transaction Server for z/OS systems
  - CICS Transaction Server for OS/390 systems
- **Note:** The external CICS interface (EXCI) uses a specialized form of MRO link to support:
  - Communication between MVS batch programs and CICS
  - Distributed Computing Environment (DCE) remote procedure calls to CICS programs

Function Shipping – Intersystem Communication (ISC)

- **ISC normally requires an SNA access method, such as VTAM, to provide the necessary communication protocols**
- **This form of communication can also be used between CICS systems in the same operating system or MVS sysplex, but MRO provides a more efficient alternative**
- **The SNA protocols that CICS uses for intersystem communication are Logical Unit Type 6.2 , which is the preferred protocol, and Logical Unit Type 6.1 which is used mainly to communicate with IMS systems**
- **CICS Transaction Server for z/OS can use ISC Function Shipping to communicate with:**
  - Other CICS Transaction Server for z/OS systems
  - CICS Transaction Server for OS/390 systems
  - CICS Transaction Server for VSE/ESA(TM)
  - CICS/VSE® Version 2
  - CICS Transaction Server for iSeries(TM)
  - CICS/400® Version 4
  - CICS on Open Systems
  - CICS Transaction Server for Windows

Function Shipping with the old Workload

- A word of caution – Issuing 100,000 File Control requests in the same transaction is more like batch work (although, applications do this very thing) This will show a huge difference between Local and Function Shipping. The numbers are included here for a reference only



Workload for Function Shipping

- DHUN Transaction – Issues 100 Direct Reads against a tuned LSR dataset
- SHUN Transaction – Issues StartBrowse and 100 ReadNext commands against a tuned NSR dataset
- Local Baselines
  - DHUN .0011 Response Time and .0009 CPU Time
  - SHUN .0065 Response Time and .0011 CPU Time

Function Shipping LU 6.2 – Different LPAR

- SHUN Transaction that Reads Sequentially 100 records
  - EXEC CICS StartBrowse followed by 100 EXEC CICS ReadNext
- DHUN Transaction that issues 100 Direct Reads

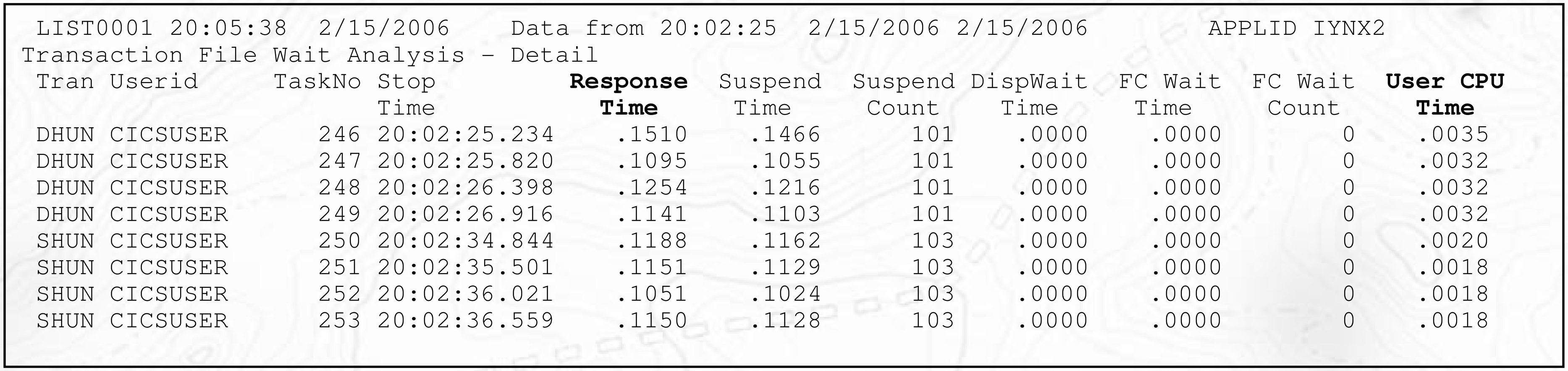
```
 LIST0001 Printed at 20:57:24  2/15/2006    Data from 20:54:42  2/15/2006        APPLID IYNX2
Transaction File Wait Analysis - Detail
 Tran Userid      TaskNo Stop         Response  Suspend  Suspend DispWait  FC Wait  FC Wait  User CPU
                         Time           Time      Time     Count     Time     Time    Count     Time
 DHUN CICSUSER       374 20:54:42.017    .1470    .1329      201    .0000    .0000        0    .0055
 DHUN CICSUSER       375 20:54:42.597    .1290    .1157      201    .0000    .0000        0    .0050
 DHUN CICSUSER       376 20:54:43.133    .1320    .1176      201    .0000    .0000        0    .0052
 DHUN CICSUSER       377 20:54:43.692    .1231    .1096      201    .0000    .0000        0    .0051
 SHUN CICSUSER       378 20:54:45.543    .1446    .1344      204    .0000    .0000        0    .0027
 SHUN CICSUSER       379 20:54:46.068    .1419    .1323      204    .0000    .0000        0    .0020
 SHUN CICSUSER       380 20:54:46.595    .1399    .1306      204    .0000    .0000        0    .0020
 SHUN CICSUSER       381 20:54:47.058    .1386    .1291      204    .0000    .0000        0    .0021
```

- DHUN Response Time went from .0011 Local to .1413
  - CPU on the AOR went from .0009 to .0050
- SHUN Response Time went from .0065 Local to .1399
  - CPU on the AOR went from .0011 to .0020

NOTE: CPU on the FOR was comparable to Local Workload

Function Shipping XCF/MRO – Different LPAR

- SHUN Transaction that Reads Sequentially 100 records
  - EXEC CICS StartBrowse followed by 100 EXEC CICS ReadNext
- DHUN Transaction that issues 100 Direct Reads

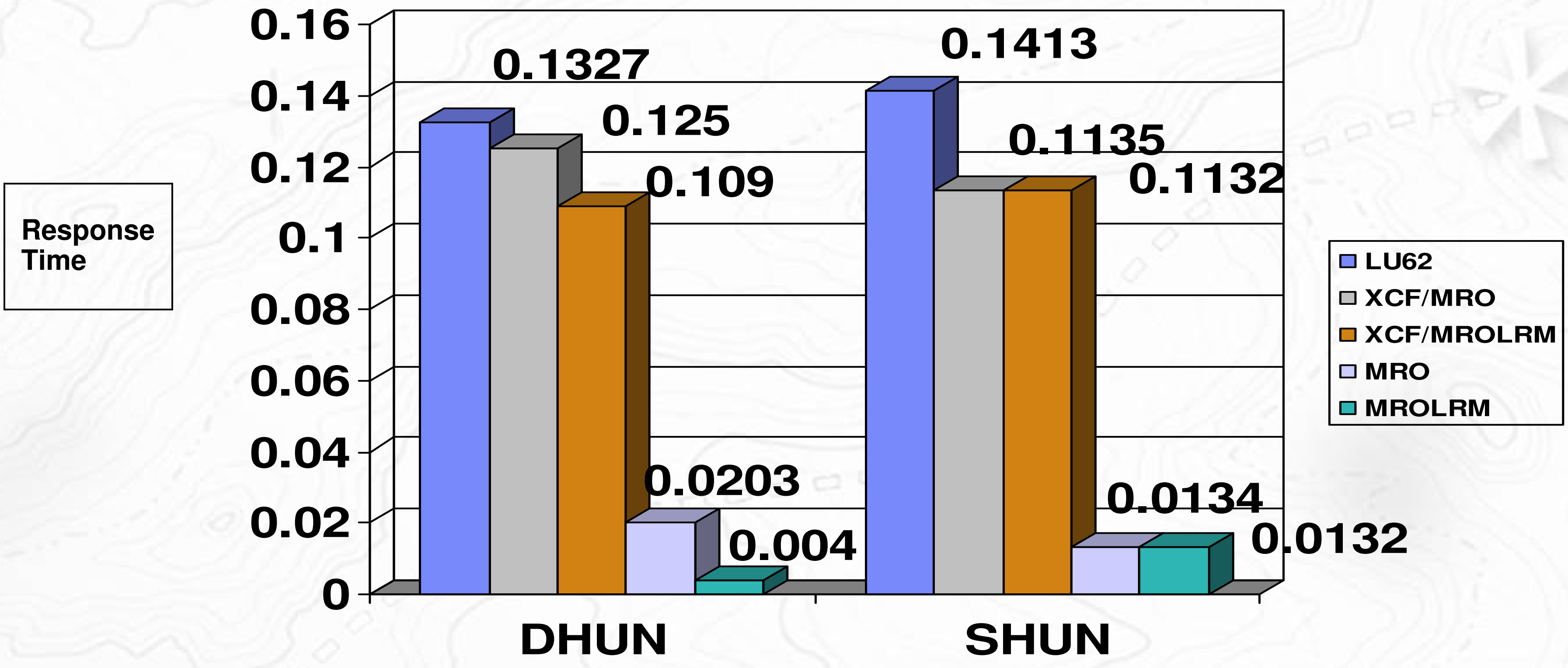


```
 LIST0001 20:05:38  2/15/2006    Data from 20:02:25  2/15/2006 2/15/2006        APPLID IYNX2
Transaction File Wait Analysis - Detail
 Tran Userid      TaskNo Stop          Response  Suspend  Suspend DispWait  FC Wait  FC Wait  User CPU
                         Time            Time      Time    Count     Time     Time    Count     Time
 DHUN CICSUSER       246 20:02:25.234    .1510    .1466      101    .0000    .0000        0    .0035
 DHUN CICSUSER       247 20:02:25.820    .1095    .1055      101    .0000    .0000        0    .0032
 DHUN CICSUSER       248 20:02:26.398    .1254    .1216      101    .0000    .0000        0    .0032
 DHUN CICSUSER       249 20:02:26.916    .1141    .1103      101    .0000    .0000        0    .0032
 SHUN CICSUSER       250 20:02:34.844    .1188    .1162      103    .0000    .0000        0    .0020
 SHUN CICSUSER       251 20:02:35.501    .1151    .1129      103    .0000    .0000        0    .0018
 SHUN CICSUSER       252 20:02:36.021    .1051    .1024      103    .0000    .0000        0    .0018
 SHUN CICSUSER       253 20:02:36.559    .1150    .1128      103    .0000    .0000        0    .0018
```

- DHUN Response Time went from .1327 to .1250 compared to LU62
  - CPU on the AOR went from .0050 to .0032
- SHUN Response Time went from .1413 to .1135 compared to LU62
  - CPU on the AOR went from .0020 to .0018

NOTE: CPU on the FOR was comparable to Local Workload

DHUN and SHUN all Runs



Note: MROLRM in the FOR will keep the mirror task around for the life of the transaction. Without MROLRM Transaction DHUN Will have the Mirror Transaction in the FOR torn down and rebuilt with each Direct Read. It does not help Transaction SHUN since a StartBrowse/ReadNext will keep the mirror transaction in the FOR active.

Record Level Sharing (RLS)

RLS

- RLS has the following benefits
  - Exploits the parallel sysplex
  - Reduces Lock Contention
    - Locks throughout the sysplex are at a record level
  - Removes FOR capacity constraints
  - Improves Availability
    - Eliminates the FOR as a single point of failure
    - An entire dataset is not taken offline for a backout failure
  - Improves sharing between CICS and batch
  - Improves Integrity
    - Full Write integrity through many updaters throughout the sysplex
    - Various forms of read integrity

Record Level Sharing

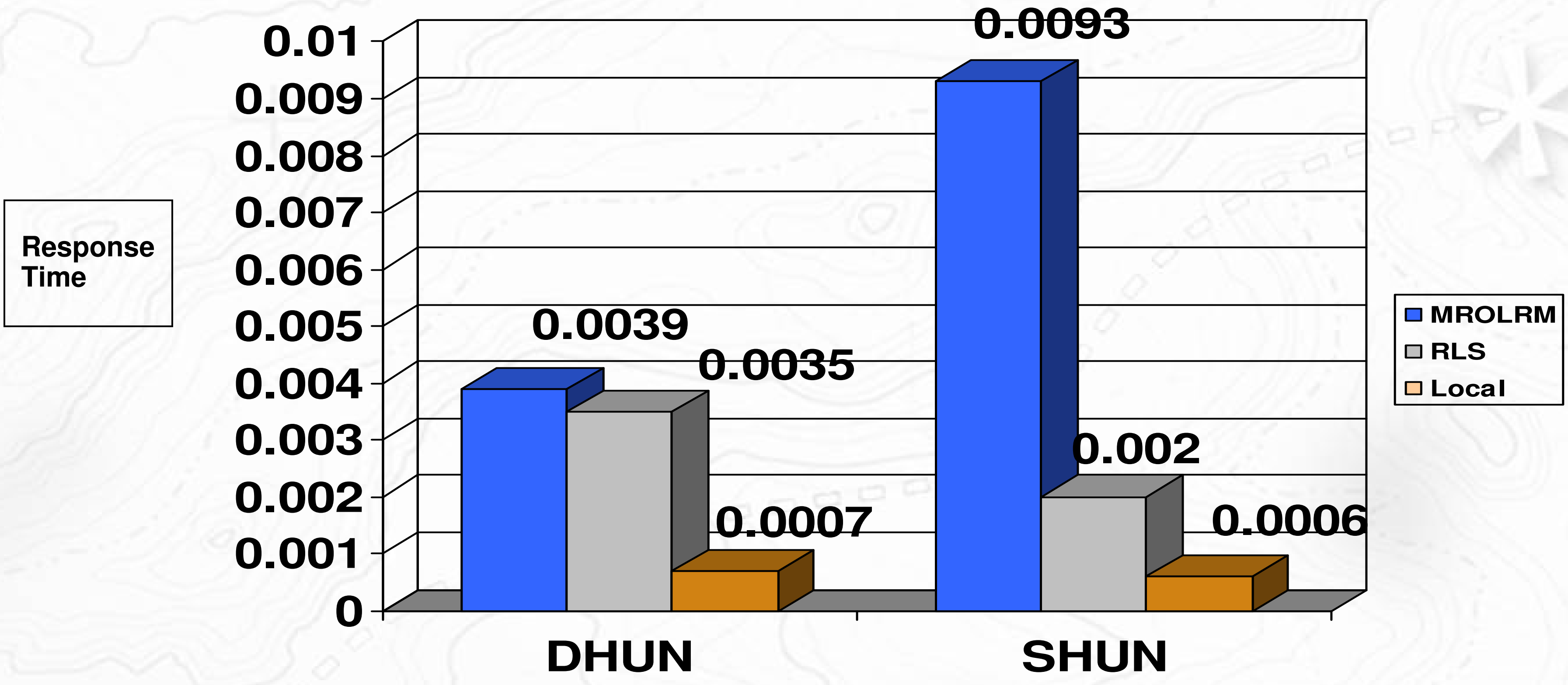
- SHUN Transaction that Reads Sequentially 100 records
  - EXEC CICS StartBrowse followed by 100 EXEC CICS ReadNext
- DHUN Transaction that issues 100 Direct Reads

```
  LIST0001 Printed at 20:02:41  2/23/2006    Data from 20:01:37  2/23/2006    APPLID IYNX7
Transaction File Wait Analysis - Detail
 Tran     Dispatch  TaskNo Stop          Response  Suspend Dispatch DispWait  FC Wait  FC Wait User CPU  RLS CPU RLS Wait RLS Wait
          Time              Time            Time     Time     Count     Time     Time    Count    Time     Time    Count     Time
DHUN     .0035     3693 20:38:41.837   .0035    .0000        1     .0000    .0000        0     .0007    .0016       0     .0000
DHUN     .0035     3694 20:38:42.078   .0035    .0000        1     .0000    .0000        0     .0007    .0016       0     .0000
DHUN     .0034     3695 20:38:42.310   .0034    .0000        1     .0000    .0000        0     .0007    .0016       0     .0000
DHUN     .0035     3696 20:38:42.547   .0035    .0000        1     .0000    .0000        0     .0007    .0016       0     .0000
SHUN     .0020     3711 20:38:47.429   .0020    .0000        2     .0000    .0000        0     .0007    .0009       0     .0000
SHUN     .0020     3712 20:38:47.634   .0020    .0000        2     .0000    .0000        0     .0007    .0010       0     .0000
SHUN     .0020     3713 20:38:47.837   .0020    .0000        2     .0000    .0000        0     .0007    .0008       0     .0000
SHUN     .0020     3714 20:38:48.041   .0020    .0000        2     .0000    .0000        0     .0007    .0008       0     .0000
```

- DHUN Response Time went from .0007 Local to .0035 using RLS
  - CPU went from .00065 Local to .0023 using RLS
- SHUN Response Time went from .00637 Local to .0020 using RLS
  - CPU went from .0010 Local to .0016 using RLS

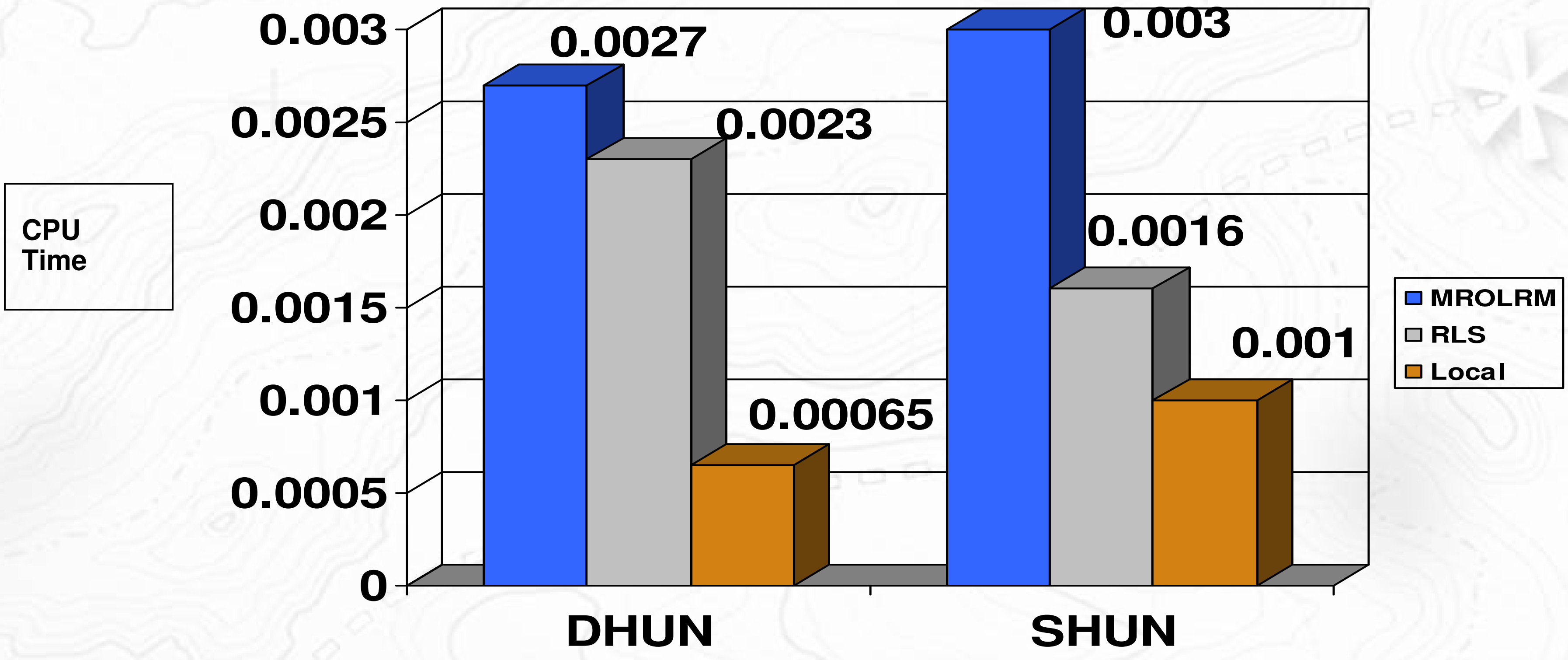
NOTE: CPU for RLS is User CPU Time + RLS CPU Time

DHUN and SHUN all Runs – Response Time



Note: MRO/XCF had a response time of .1050 for DHUN and .1022 for SHUN. Comparisons were not made for LU62 Function Shipping

DHUN and SHUN all Runs – CPU Time



Note: MRO/XCF had CPU time of .0043 for DHUN and .0042 for SHUN. Comparisons were not made for LU62 Function Shipping

RLS Update Workload One

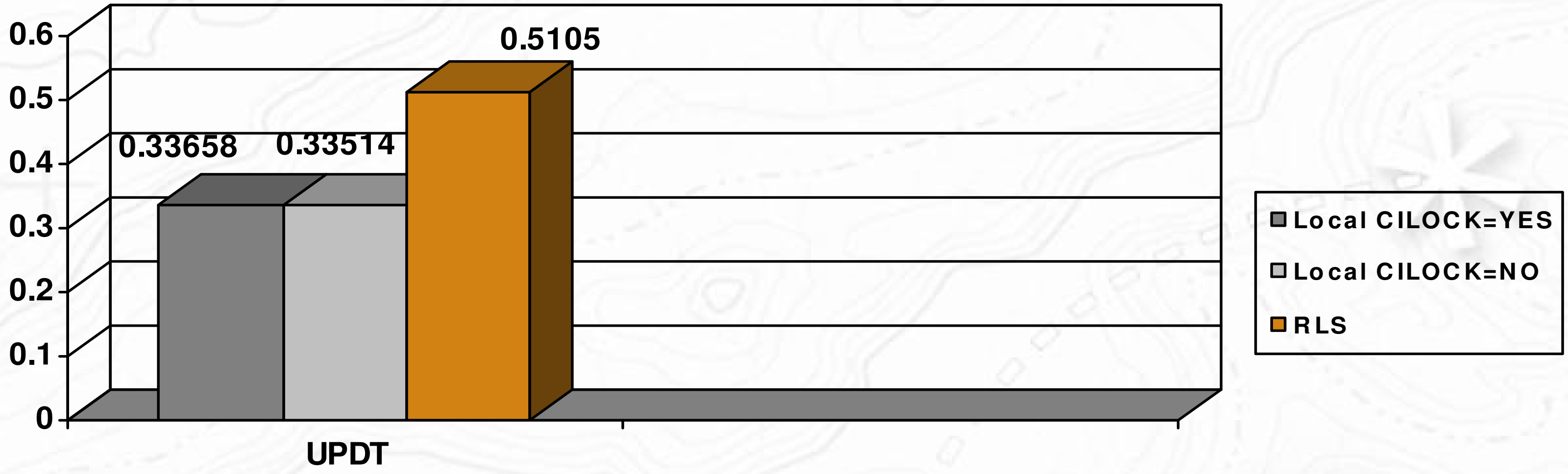
- UPDT Transaction issues 1000 Read Update / Rewrite commands against a well tuned dataset

- SUPD EXEC CICS START 50 UPDT Transactions

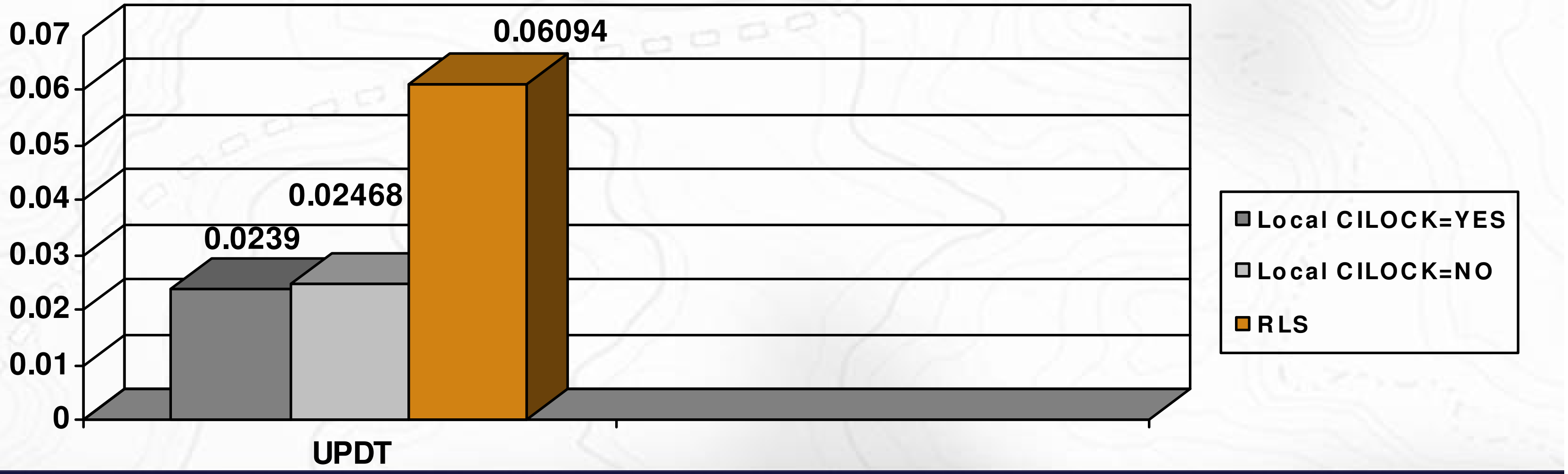
- Compared LOCAL CILOCK=YES CILOCK=NO and RLS
  - CILOCK=YES lets VSAM Lock on the Control Interval level
  - CILOCK=NO mimics Record Level Sharing on a CICS level
    - CICS will issue a record lock after the Read Update to lock other users from updating that particular record. Other transactions can then update records in the same Control Interval

UPDT TRANSACTION

Response Time

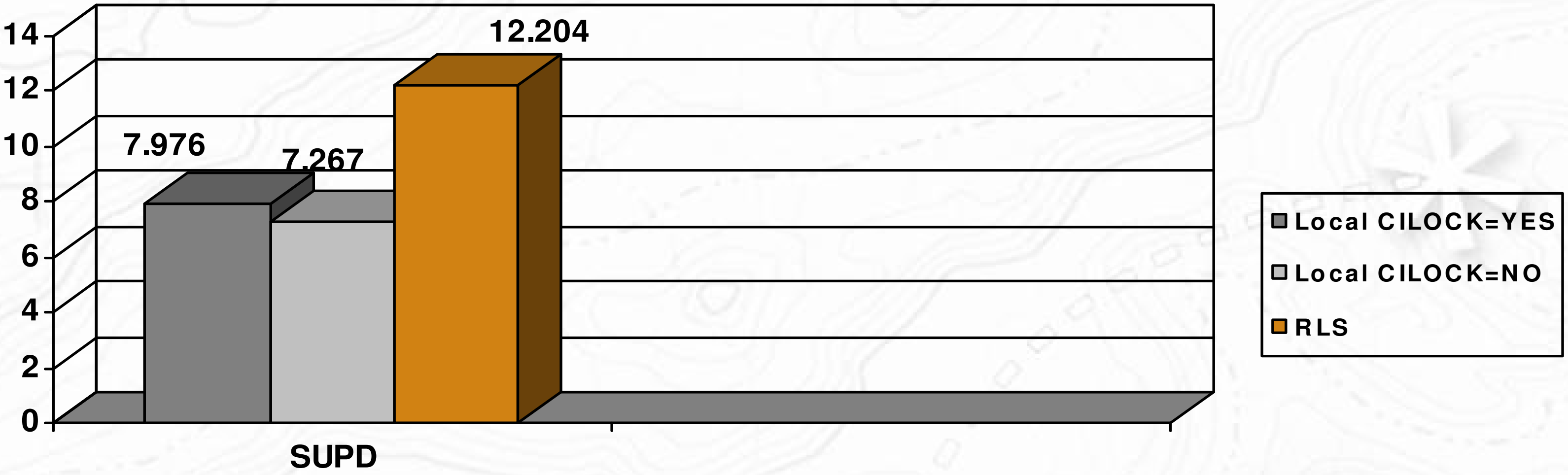


CPU Time

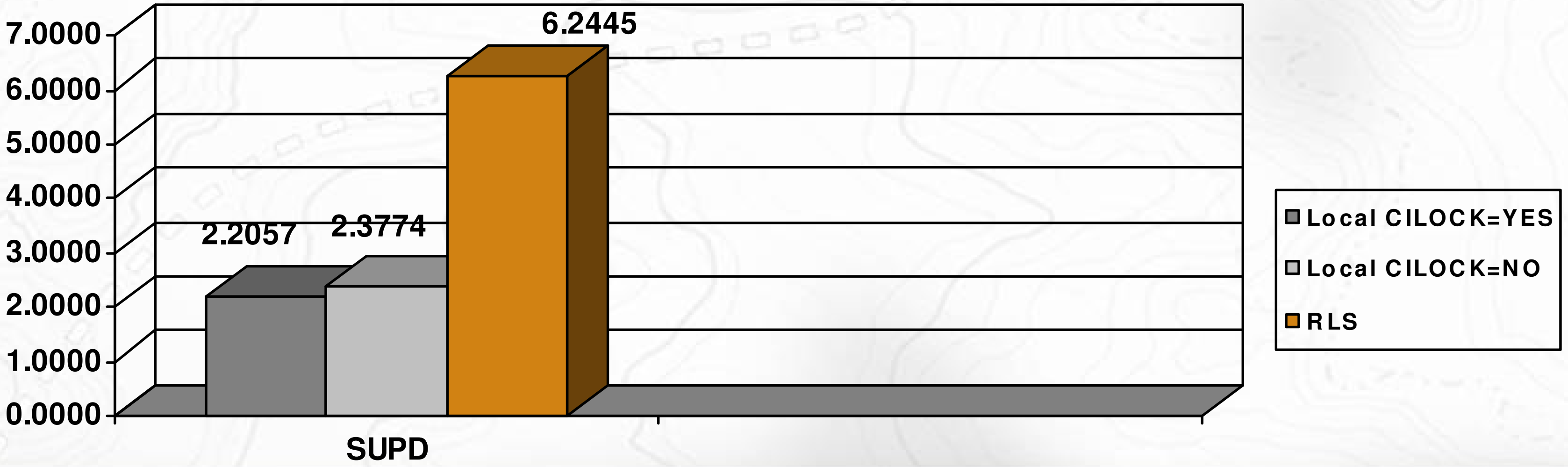


SUPD TRANSACTION

Response Time



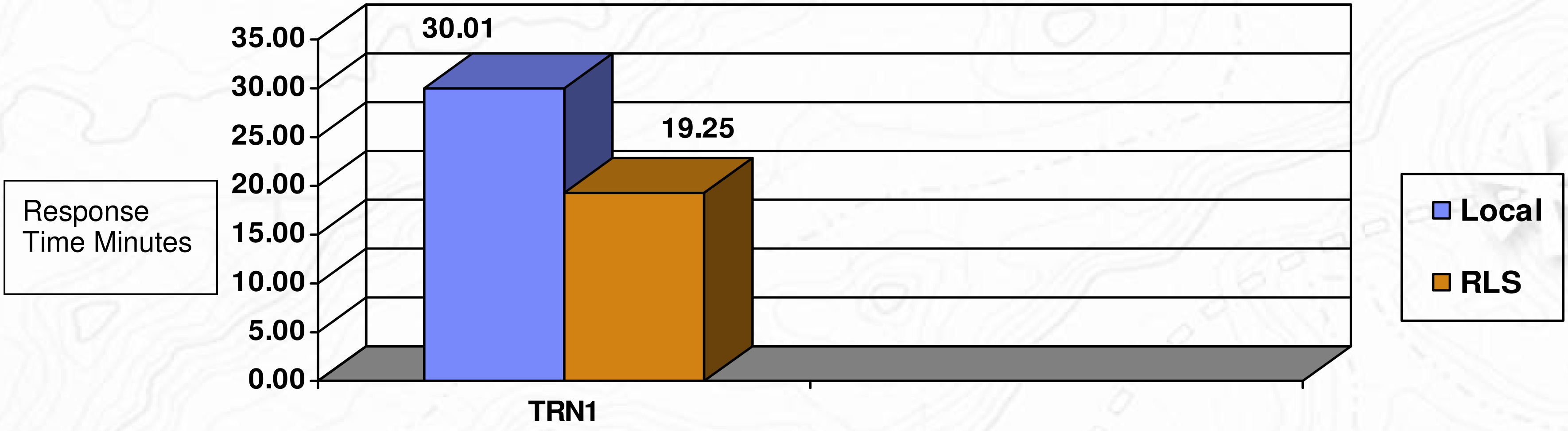
CPU Time

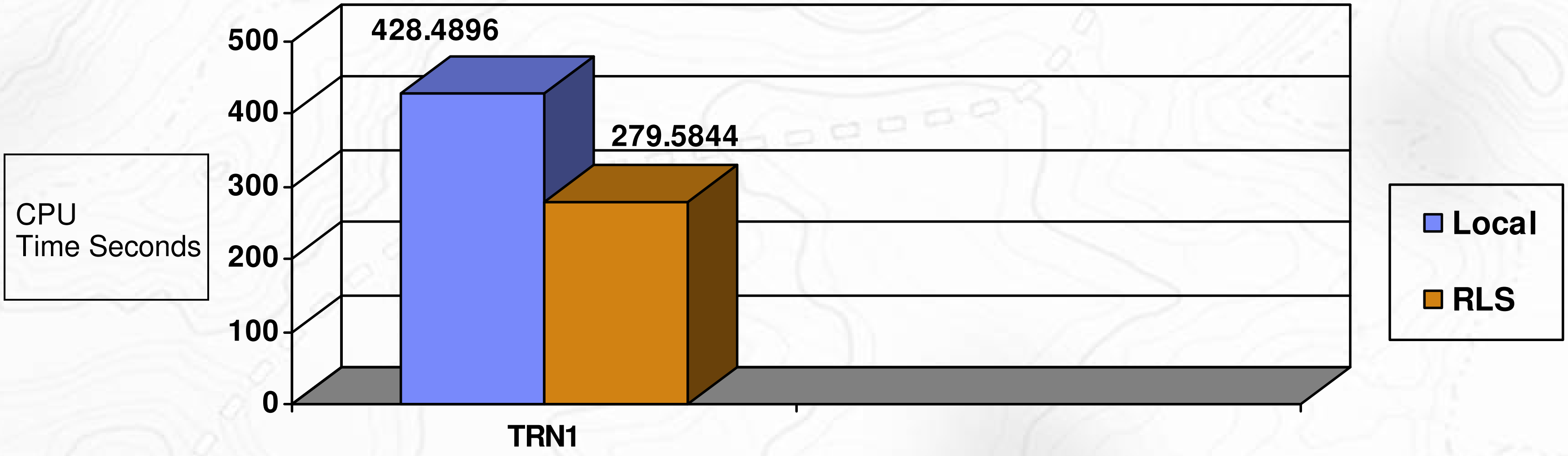


RLS Update Workload Two

- Customer workload that reads a records from a KSDS using TRN1 and EXEC CICS STARTs TRN2 to write the record to another KSDS
- Five Alternate Indexes above the output KSDS
- Input KSDS has 300,000 records for the initial iteration
  - –14,000 records for the second iteration
  - –2,200 records for the third iteration
- Used CILOCK=NO
  - –Does not make a difference in Write requests.
  - –No Records Locks are issued by CICS for Direct Writes

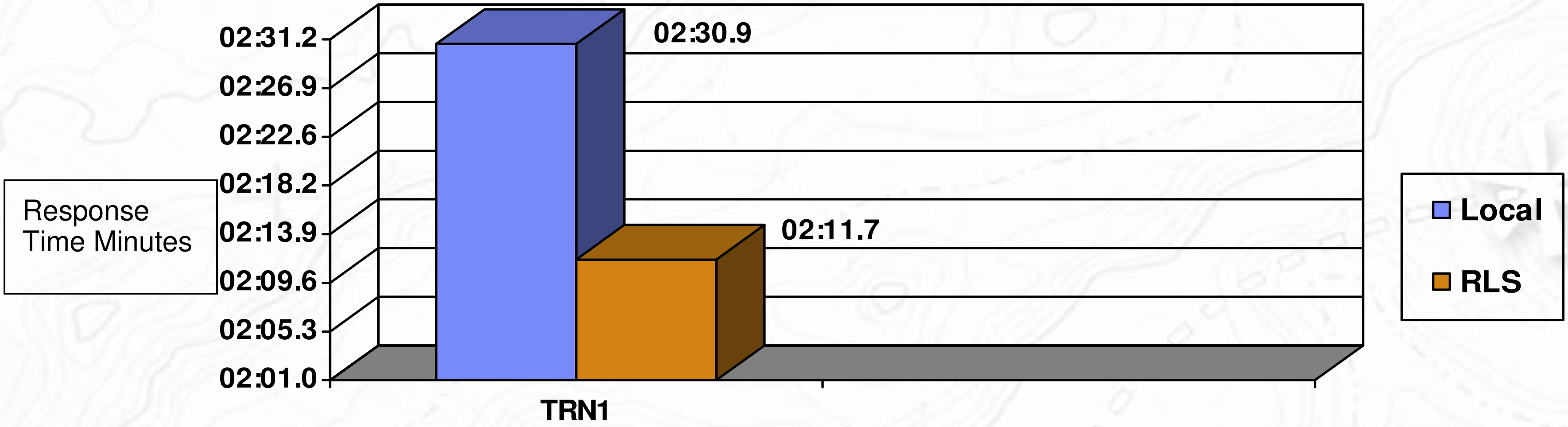
TRN1 TRANSACTION 300,000 Records

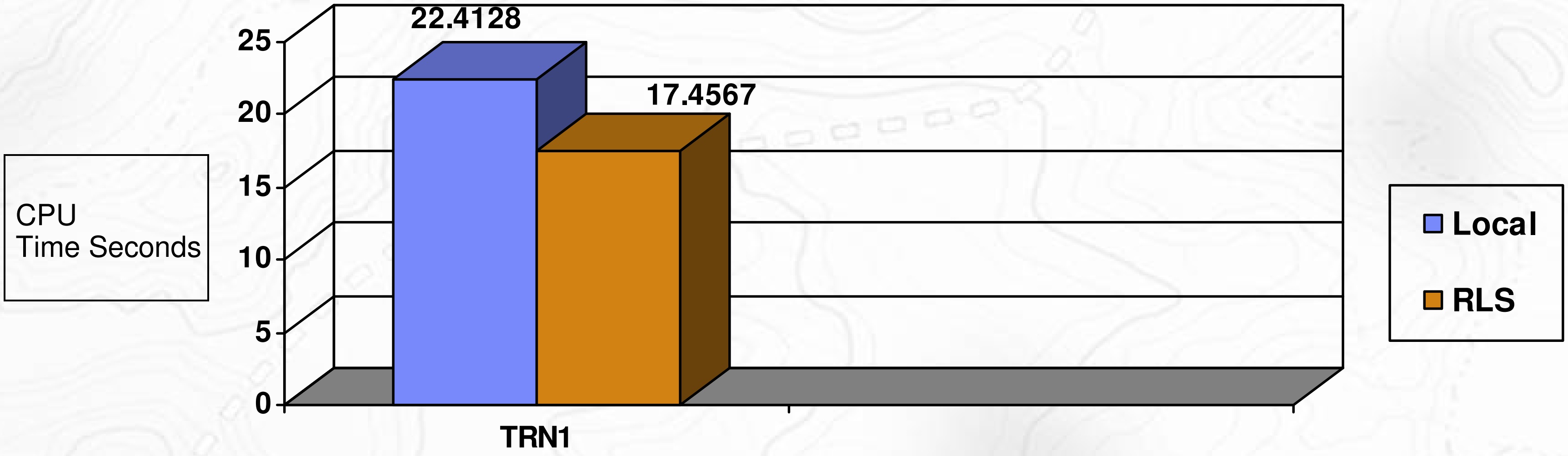




NOTE: There were 2,657,398 Exclusive Control Conflicts on the Output Dataset

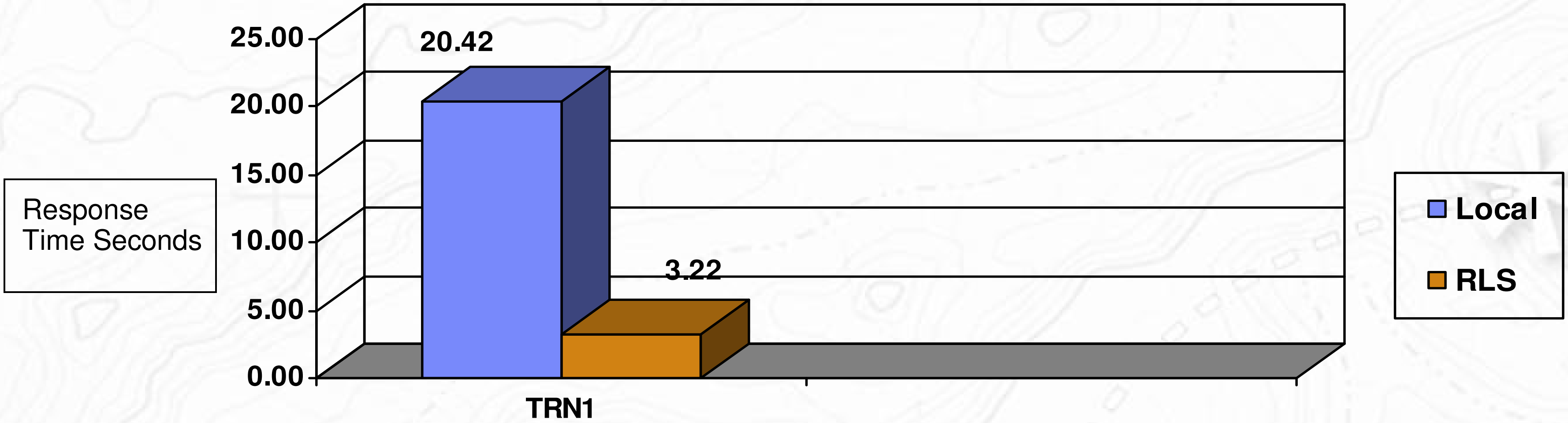
TRN1 TRANSACTION 14,000 Records

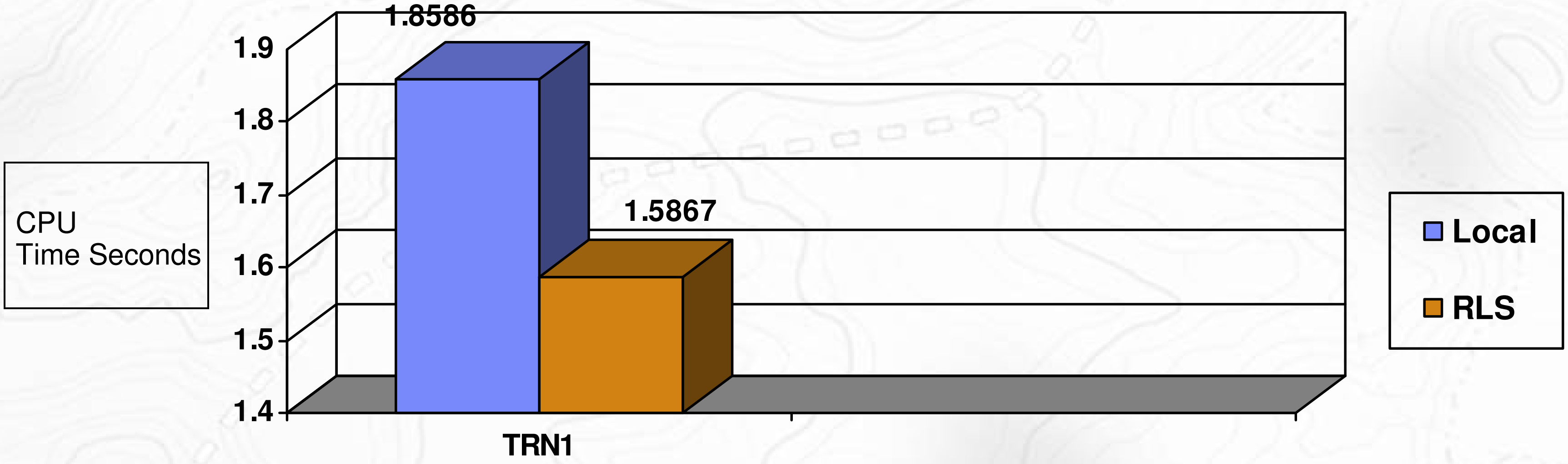




NOTE: There were 127,657 Exclusive Control Conflicts on the Output Dataset

TRN1 TRANSACTION 2,200 Records





NOTE: There were 23,614 Exclusive Control Conflicts on the Output Dataset

Summary

- **NSR**
  - Great for Sequential Read or Write Processing
  - Not a good candidate for Direct Processing
- **LSR**
  - Great for Direct Processing
  - Good for Sequential Processing
- **Shared DataTable**
  - Best candidate for Read and Browse Transactions
- **Function Shipping**
  - LU62 – High Response Time
  - XCF/MRO use instead of LU62 if in the same sysplex
  - MRO use if CICS Regions are in the same LPAR
    - Use MROLRM for Direct Request Applications
- **RLS**
  - CPU and Response time better than MROLRM even when CICS Regions are in the same LPAR
  - CPU and Response time could be better than Local depending on the workload

Questions
and
Answers

eSupport Update

- See Scotty at Product Expo for a live demo of the CICS Support Web Site: http://www.ibm.com/software/htp/cics/tserver/support/
- NEW "Fixes by version"
  - APAR tables summarizing all the fixes for each version of CICS TS for z/OS
  - Link to or search for 7008833
- Featured documents for CICS TS
  - Technotes identified as most valuable
  - Link to or search for 7006900
- eSupport Knowledge Base (Web content) is now part of IBMLink 2000 SIS search

.