Telcordia Technologies

Performance from Experience

IMS Technical Conference

IMS and LE Experiences at Telcordia

Session C05

Steve Nathan

stephen.nathan@telcordia.com

Disclaimer

- The purpose of this presentation is to provide a technical perspective of Telcordia's experience using IMS and LE.
- Although this document addresses certain IBM products, no endorsement of IBM or its products is expressed, and none should be inferred.
- Telcordia also makes no recommendation regarding the use or purchase of IMS or LE products, any other IBM products, or any similar or comparable products.
- Telcordia does not endorse any products or suppliers. Telcordia does not recommend the purchase or use of any products by the participants.
- Each participant should evaluate this material and the products himself/herself.

Acknowledgements

- This presentation was prepared by:
  - Terry Seibert
    - IBM Global Services
    - tgseiber@us.ibm.com
  - Avri Adleman
    - Telcordia Technologies
    - aadleman@telcordia.com
- They have spent MANY hours studying this topic and working with IMS and LE development to make this environment work

Telcordia Technologies

Performance from Experience

Trademarks

- The following terms are trademarks of the IBM corporation in the United States or other countries or both:
  - C/370
  - IBM
  - IMS
  - Language Environment
  - Open Edition
  - OS/390
- UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited
- Other company, product, and service names may be trademarks or service marks of others

Presentation Outline

- Overview – What is LE?
- Migrating to LE
- Runtime Options
- Debugging LE

Introduction

- This session will cover Language Environment (LE) setup and options that pertain to an IMS environment
- Topics will include runtime and initialization options and any differences in setting up IMS online, BMP and batch environments
- The new IMS Version 8 Dynamic Runtime Options will also be discussed
- This presentation was prepared at the OS/390 V2R10 level
  - Other releases are referenced
- This presentation will make no attempt to discuss applications which may also use OS/390 UNIX Systems Services

LE Overview

- What is Language Environment (LE)?
  - Single runtime environment for High Level Languages
    - Basic support routines
      - Init/term, storage, messages, conditions
    - Callable Services
      - Date, time, etc
    - Language specific routines
      - C/C++
      - Cobol
      - PL/I
      - Fortran

LE Overview

- What is Language Environment (LE)?
  - Process
    - Address space
  - Enclave
    - Main program and called subroutines
  - Thread
    - Task

LE Overview

- Why use LE?
  - Base element of OS/390 & z/OS
  - Prerequisite for applications built with newer compilers
  - Replaces obsolete/stabilized runtime library products

Migration – Using Multiple LE Releases

- LE is upward compatible
  - Applications built on one level of LE will continue to run on later releases of LE without the need to relink or recompile
- New in OS/390 R10 – LE is also downward compatible
  - You may develop applications on higher releases of LE for use on platforms running lower releases of LE
  - The LE Programming Guide lists guidelines and restrictions
    - This is NOT a rollback of new function to prior releases
    - The system used to build the applications must be at least OS/390 V2R10
  - Toleration PTFs for lower OS/390 releases are in a PSP bucket
    - Upgrade OS390R10 subset LANGENV
    - Not in z/OS

Migrating IMS Regions to LE

- Make sure you & your applications are ready
  - Read the language-specific LE Migration Guides
    - LE guide
    - Language specific guide
  - PLAN, PLAN, PLAN
    - Know current runtime options
  - Perform regression tests
    - Include error scenarios
- Make sure your Vendor tools are LE enabled
  - There is a list in the LE Migration Guide – Appendix A or call the vendor

Migrating IMS Regions to LE

- LPA, LNKLIST or STEPLIB for LE modules?
  - LNKLIST for most LE modules
    - SCEERUN (PDS) and SCEERUN2 (new PDSE – V2R10)
  - LPA for heavily used LE modules
    - SCEELPA contains LPA eligible LE modules
    - Also check language-specific recommendations in Migration Guides
  - See OS/390 Program Directory
    - LNKLSTxx Considerations
  - APAR II10425
    - How to install OS/390 without LE in the LNKLIST

Migrating IMS Regions to LE

- STEPLIB for LE modules
  - Use STEPLIB to test LE
    - CEE.SCEERUN and CEE.SCEERUN2
  - Use STEPLIB until LE migration is complete
  - There are considerations for IMS preload

PLICALLA

- If your load module is using PLICALLA (as many IMS programs do)
  - In linked steps you must do one of the following:
    - Put SIBMCALL or SIBMCALL2 ahead of SCEELKED
    - Explicitly INCLUDE LE-provided PLISTART CSECT
- If you load module is not using PLICALLA
  - Do not do either of the above because they will needlessly increase you load module size

IMS Data Capture Exit

- The IMS Customization manual says:
  - "IMS does not support exit routines running under Language Environment for OS/390"
- IMS Data Capture Exits can be written high-level languages
  - And soon you will have no option but to run under LE
- IBM has tested this environment and will now support it
  - The manuals will be updated
  - Fixes will be required

IMS Data Capture Exit

- OS/390 R10 or above requires an IMS APAR
  - DFSPC40 must initialize the LINKX parameter list
  - PQ47638 (V5) – PQ46980 (V6) – PQ47639 (V7)
- APARs PQ35776 and PQ31566 document Abendu4087 with F1SA in Register 2 after AbendU4000 in IGZCFCC
  - These APARs were closed “CAN”
    - Use the ABPERC(U4000) runtime option to percolate the U4000
    - Tailor LE assembler exit CEEBXITA to set the runtime option

Library Retention Routine

- Library Retention Routine (LRR)
  - Keeps LE resources in memory for better performance
  - Can not be used for application programs
    - Use IMS Preload for that
    - Uses LE PREINIT
- LRR setup
  - Specify CEELRRIN in the DFSINTxx member of the IMS PROCLIB
  - Specify ‘xx’ as the suffix on the PREINIT keyword in the IMS Dependent Region JCL

LRR Storage Tuning User Exit

- The LRR Storage Tuning User Exit has two functions
  - Collect LE storage tuning information without having to run with the RPTSTG option
  - Set the LE runtime options STACK, LIBSTACK, HEAP, ANYHEAP, and BELOWHEAP for each LE enclave
- The exit name must be CEEBSTX (for non-CICS with LRR)
- There is a sample in SCEESAMP named CEEWBSTX
- See the LE Customization manual for details

LRR Load Notification User Exit

- The LRR Load Notification User Exit can be used to improve performance by preventing the use count for frequently used modules from dropping below one
  - Invoked at region initialization
  - Invoked after each successful load
    - Can issue a second load to increase the use count
  - Invoked at region termination
    - Can issue a delete to lower the use count to zero
- Exit name is CEEBLNUE and there is a sample of the same name in SAMPLIB
- See the LE Customization manual for details

LE Runtime Options

- There are MANY LE runtime options
- They are documented in the LE Programming Reference manual
- The LE Migration Guide lists current recommendations
  - Language specific
  - Mixed language applications
  - CICS environments
    - For some reason CICS always seems to be an exception
  - Non-CICS environments
    - This includes IMS

LE Runtime Options

- ABTERMENC
  - ABTERMENC sets the enclave termination behavior for an enclave ending with an unhandled condition of severity 2 or greater
  - TRAP(ON) must be in effect for ABTERMENC to have an effect
  - Valid values are RETCD or ABEND
  - ALWAYS specify ABEND for IMS
    - This is the default starting with OS/390 V2R9

LE Runtime Options

- DEPTHCONDLMT
  - DEPTHCONDLMT specifies the extent to which conditions can be nested
  - The default is 10
  - The recommendation is 0
    - This allows an unlimited depth of condition handling
    - This also provides PL/I compatibility

LE Runtime Options

- ERRCOUNT
  - ERRCOUNT specifies how many conditions of severity 2, 3, or 4 can occur per thread before the enclave terminates abnormally
  - After the number specified in ERRCOUNT is reached, no further Language Environment condition management, including CEEHDLR management, is honored.
  - The default starting with OS/390 V2R6 is zero
  - Zero is the recommedation

LE Runtime Options

- TERMTHDACT
  - TERMTHDACT sets the level of information that is produced when Language Environment percolates a condition of severity 2 or greater beyond the first routine's stack frame
  - The default option is TRACE
    - LE generates a message indicating the cause of the termination and a trace of the active routines on the activation stack as well as an options report
  - The UADUMP option and a DD statement will get a U4039 dump
  - See the LE Programming Reference manual for all of the options and their meanings

LE Runtime Options

- TRAP
  - TRAP specifies how Language Environment programs handle abends and program interrupts
  - This option is similar to the STAE | NOSTAE runtime option currently offered by COBOL, C, and PL/I, and the SPIE | NOSPIE option offered by C and PL/I
    - But not really
  - TRAP(ON) must be in effect for the ABTERMENC runtime option to have effect

LE Runtime Options for Performance

- ALL31
  - ALL31 specifies whether an application can run entirely in AMODE 31 or whether the application has one or more AMODE 24 routines
  - This option does not implicitly alter storage, in particular storage managed by the STACK and HEAP runtime options
  - However, you must be aware of your application's requirements for stack and heap storage, because such storage can potentially be allocated above the line while running in AMODE 24
  - It is recommended that ALL31 have the same setting for all enclaves in a process
    - LE does not support the invocation of a nested enclave requiring ALL31(OFF) from an enclave running with ALL31(ON) in non-CICS environments.

LE Runtime Options for Performance

- ANYHEAP, BELOWHEAP, HEAP
  - ANYHEAP and HEAP are used by LE
  - HEAP is used by the application
- LIBSTACK, STACK (Save Areas)
  - LIBSTACK is used by LE
  - STACK is used by the application
- RPTOPTS, RPTSTG
  - Use RPTSTG suggested values to minimize GETMAINs
  - Avoid generating reports during production!!!

LE Runtime Options for Performance

- LE runtime options changed at z/OS 1.2
  - ALL31(ON)
    - Tell LE that no application routines are AMODE 24
  - STACK(,,ANY,,,)
    - Puts stack storage above the line
  - THREADSTACK(,,,ANY,,,)
    - Puts thread stacks above the line for multi-threaded applications
  - STORAGE(,,,0K)
    - Eliminates below the line reserved stack storage
- This is known as the Favor 31-Bit Application Enhancement

LE Runtime Options for Performance

- Favor 31-Bit Application Enhancement
  - IMS applications compiled with C/370 and linked with the pre-LE CTDLI stub and run with ALL31(OFF) may abend because LWS (Library Work Space) storage is not allocated
    - This is fixed with APAR PQ56143
  - The Reserve Stack needs to be a minimum of 32K
    - STORAGE(,,,nK)
    - Used by LE to process out-of-storage conditions

Setting LE Runtime Options

- There are MANY ways to set LE runtime options
  - CEEDOPT
  - CEEROPT
  - CEEUOPT
  - Load Module
  - IMS V8 Dynamic LE runtime options
  - LRR Storage Tuning User Exit

Setting LE Runtime Options

- CEEDOPT
  - Installation-wide LE default options
- CEEROPT
  - Region-wide LE options (if IMS with LRR)

Setting LE Runtime Options

- CEEUOPT
  - Application specific LE options
  - Must be linked with the application

Setting LE Runtime Options

- Load module
  - PL/I main
    - PLIXOPT
  - C main
    - #pragma runopts()
- LRR Storage Tuning User Exit
  - This was previously discussed

Setting LE Runtime Options

- IMS V8 Dynamic LE Runtime Options
  - Allows LE runtime options to be dynamically changed without updating CEExOPT and/or recompiling the application and/or recycling the IMS dependent region
  - Requires new IMS V8 Operations Manager (OM) and TSO Single Point of Control (SPOC)
  - Uses DFSBXITA, and IMS specific version of CEEBXITA
  - The setting of LE options can be dynamically added, updated, deleted, and queried
  - Requires activation in DFSPBxxx

Setting LE Runtime Options

- IMS V8 Dynamic LE Runtime Options
  - Filters are used to decide when to set the dynamic LE runtime options
    - Transaction Code
    - LTERM
    - Userid
    - Program

Debugging With LE

- ABEND codes are different with LE
  - Why be consistent?!?!?
  - Most LE abends are U4038/U4039
    - About as useful as IMS U4095
- Debug using error messages – not abend codes
  - e.g. Abend0C4 becomes message CEE3204S
- The MSGFILE runtime option species the DDNAME for all runtime diagnostics and reports generated by RPTOPTS and RPTSTG
  - The default is SYSOUT

Debugging With LE – Dump Files

- CEEDUMP
  - Formatted dump of LE storage/data
  - Content depends on TERMTHDACT() suboption
- CEESNAP
  - Application generated dump information
- SYSUDUMP
  - If TRMTHDACT(UADUMP) and SYSUDUMP DD card
  - Formatted dump but no formatting of LE information
- SYSMDUMP
  - If TRMTHDACT(UADUMP) and SYSMDUMP DD card
  - Use when reporting problems to IBM
  - IPCS verbexit LEDATA/CEEERRIP formats LE data

Debugging With LE – Control Blocks

- Common Anchor Point (CAA)
  - Pointed to by Register 12
- Stack Frame/Dynamic Save Area (DSA)
  - Pointed to by Register 13
  - DSA's are backchained at DSA+4
- Condition Information Block
  - CEECAA+x'2D8' points to current CIB
- Machine State Information Block (ZMCH)
  - Pointed to by CIB+x'24'

Debugging with IMS and LE

- IMS & LE do coordinate condition handling!
  - If an error occurs in an IMS environment LE will send the condition to IMS
- There are a number of APAR's dealing with IMS and LE
  - Some have been documented in this presentation
  - Others can be found be searching IBMLINK
    - This is HIGHLY recommended

Uninitialized Variables

- Prior to LE uninitialized variables had a “high probability of being binary zero
  - Many programs relied on this
- With LE many uninitialized variables contain “garbage”
  - LE gets the storage and uses it for initialization and then uses it for the application
- This was the source of MANY (MANY MANY) abends and unexpected conditions and logic errors

Conclusion

- Implementing LE in an IMS environment requires hard work
- Plan by reading the Migration manuals
- Review runtime options before migration
- Consider LRR for performance
- Check for uninitialized variables
- Do extensive testing
  - Including error scenarios

Questions?

Telcordia Technologies

Performance from Experience

RETURN TO INDEX