Die Optionen von Language Environment ™ unter dem Gesichtspunkt der Performance

Version 03a

Stand:
14. Juli 2003

Layout überarbeitet:
18. September 2013

Eine Ausarbeitung von:

cps4it
Ralf Seidler
Stromberger Straße 36A, 55411 Bingen
Fon: 06721-992611 • Fax: 06721-992613 • Mail: ralf.seidler@cps4it.de
www.cps4it.de

Ein Auftrag der
Firma

contigon
informationstechnologie + consulting gmbh
Mehrbachstrasse 4, 65321 Heidenrod

Inhaltverzeichnis

1 Zu diesem Dokument

1.1 Ziel des Dokuments

Dieses Dokument beschreibt die Optionen von Language Environment™ (LE), die einen merklichen Einfluss auf das Laufzeitverhalten von Anwendungs- und Systemprogrammen haben. Aus den Ergebnissen sollen Firmen, welche die Laufzeitkomponente LE der Firma IBM im Einsatz haben, Folgerungen ziehen können, um die Optionen performanceoptimal einzustellen. Die Voraussetzungen für den Einsatz der „optimalen“ Optionen werden, soweit dies erforderlich ist, beschrieben. Das Dokument stellt eine zusätzliche Hilfe dar, welche die vorhandene Literatur der IBM™, vornehmlich Bockmanager-Broschüren und Vorträge zum Thema LE, unter dem Blickwinkel der Performance zusammenfasst.

1.2 Garantie

Diese Ausarbeitung ist nach bestem Wissen und Gewissen entstanden. Es kann aus Gründen der Haftung keine Garantie irgendwelcher Art gegeben werden.

1.3 Voraussetzungen

Sollten Voraussetzungen in irgendeiner Form vorhanden sein, werden diese im Dokument ausdrücklich angeführt.

1.4 Literatur und Änderungen

Auf weiterführende Literatur bzw. Quellenangaben wird im Anhang hingewiesen. Änderungen gegenüber einer vorherigen Version dieses Dokuments sind am Rand gekennzeichnet.

1.5 Aufbau des Dokuments

Jede Option, die merkliche Auswirkungen auf die Performance hat, wird wie folgt erläutert:

a) Die Syntax der Option
b) Beschreibung der Option
c) Die aus Sicht der Performance optimale Einstellung
d) Die empfohlene Einstellung
   Hierbei ist zu beachten, dass es manchmal sinnvoll sein kann, nicht die performanteste Einstellung zu wählen.
e) Die Default Einstellung
f) Voraussetzung(en) für die empfohlene bzw. optimale Einstellung.
g) Eventuelle Anmerkung(en)

Optionen, die mit Speicherbereichen zu tun haben, werden in einem separaten Kapitel behandelt. Wegen der Übersichtlichkeit wurde auch zwischen CICS und non-CICS getrennt, obwohl viele Optionen gleich behandelt werden können. Das letzte Kapitel gibt Vorschläge für Vorgehensweisen, um eine optimale LE-Umgebung zu erhalten.

2 Sonstige Vorbemerkungen

2.1 Ansprechpartner

Für Rückfragen, Anregungen und Kritik wenden Sie sich an die auf dem Deckblatt genannte(n) Adresse(n).

2.2 Allgemeines

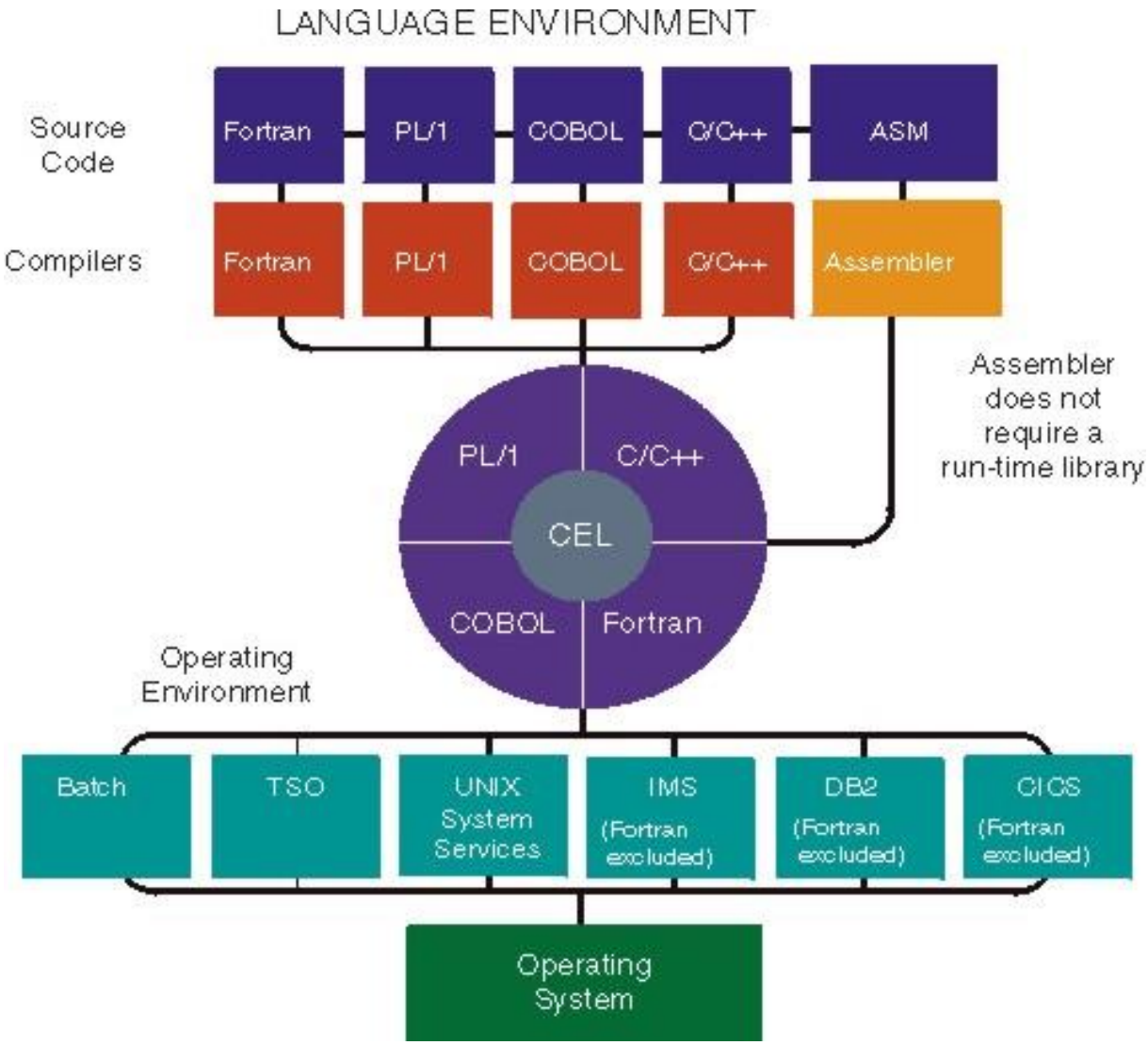
IBM's Language Environment for OS/390 & VM (Language Environment™) bietet eine einzige Run-Time Umgebung für C, C++, COBOL, Fortran, PL/I und Assembler Anwendungen. Die „common library" von LE beinhaltet gemeinsame Services wie Message-, Datum- and Zeit-Funktionen, mathematische Funktionen, Utilities, System Services und Unterstützung von Subsystemen. Alle diese Services sind aufrufbar über Interfaces, die unabhängig von der rufenden Sprache sind. Die Interfaces sind entweder direkt aufrufbar oder auch indirekt über sprachabhängige Services. All dies garantiert konsistente und vorhersagbare Ergebnisse für die Anwendungen unabhängig von der Sprache, in der sie geschrieben ist.

2.3 Voraussetzungen

Im Lauf der Zeit hat die Firma IBM im Hinblick auf die Performance in und unter LE sehr viele Verbesserungen auch über PTFs für LE, aber auch für anderes OS/390-Subsysteme, getätigt. Daher bezieht sich diese Ausarbeitung nur auf die neueren Releasestände von LE ab der Version, die unter OS/390 R 2.9 oder höher eingesetzt wird. Dies entspricht dem LE-Level 0C oder höher. Zu beachten ist, dass der Level 09, passend zu OS/390 R2.6, nur noch bis zum 31. März 2002 supported wird.
Empfohlen wird der Einsatz von (mindestens) OS/390 LE 2.10, da ab diesem Release auch eine Abwärtskompatibilität unterstützt wird, sofern die compatibility toleration APARs PQ 30805 und PQ 33358 eingesetzt werden.
Selbstverständlich gelten die Aussagen ebenfalls für z/OS V1R2.0 Language Environment und spätere heute schon freigegebene Releases. An die z/OS Version lehnt sich die angegebene Syntax an. Die Run-Time Optionen ändern sich teilweise von einem Release zum anderen, sodass die Syntax angepasst werden muss.

Informationen über Optionen, die nur die Sprachen FORTRAN, C/C++, PL/I betreffen, werden in diesem Dokument nicht berücksichtigt.

2.4 Schaubild



3 non-CICS Optionen - ohne Speicherparameter

3.1 AIXBLD

a) Syntax
AIXBLD/NOAIXBLD
b) Beschreibung
Zugriffsoptimierung bei VSAM-Dateien mit einem Alternativ-Index.
c) Performance optimal
NOAIXBLD
d) empfohlen
NOAIXBLD
e) Default
NOAIXBLD
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Bei Anwendungen, die intensiv VSAM-Dateien mit Alternativ-Index bearbeiten, ist es sinnvoll, über einen Einsatz von AIXBLD nachzudenken. Diese Option sollte dann auf Ebene der Anwendung gesetzt werden.

3.2 CHECK

a) Syntax
CHECK=((ON/OFF),OVR/NOOVR)
b) Beschreibung
Während der Ausführungszeit werden Fehler in einer COBOL-Anwendung geflaggt, die sich auf Indices, Subscripte und Referenzmodifikationen beziehen.
c) Performance optimal
CHECK=((OFF),OVR)
d) empfohlen
CHECK=((ON),OVR)
e) Default
CHECK=((ON),OVR)
f) Voraussetzung(en)
COBOL-Anwendung
g) Anmerkung(en)
Nach der IBM-Literatur hat die Option CHECK=ON keinerlei Auswirkung auf die Performance, wenn die Anwendung mit NOSSRANGE umgewandelt worden ist. Untersuchungen haben zwar das Gegenteil gezeigt, trotzdem wird CHECK=ON empfohlen. Gleichzeitig sollte aber die COBOL-Compile-Option NOSSRANGE aktiviert sein. In Ausnahmefällen darf SSRANGE kurze Zeit benutzt werden.

3.3 DEBUG

a) Syntax
DEBUG / NODEBUG bzw. DEBUG=((ON/OFF),OVR/NOOVR)
b) Beschreibung
DEBUG aktiviert die COBOL Batch DEBUG Facility, die in den Declaratives (USE FOR DEBUGGING) gesetzt worden ist.
c) Performance optimal
NODEBUG bzw. DEBUG =((OFF),OVR)
d) empfohlen
NODEBUG bzw. DEBUG =((OFF),OVR) für Produktion, NODEBUG bzw. DEBUG =((ON),OVR) für Test
e) Default
NODEBUG bzw. DEBUG =((OFF),OVR)
f) Voraussetzung(en)
Die Option gilt nur für COBOL-Anwendungen.

3.4 PROFILE

a) Syntax
PROFILE=((ON/OFF,' '),OVR/NOOVR)
b) Beschreibung
PROFILE kontrolliert einen optionalen Profiler, der Performance Daten der Anwendung sammelt.
c) Performance optimal
PROFILE=((OFF,' '),OVR)
d) empfohlen
PROFILE=((OFF,' '),OVR)
e) Default
PROFILE=((OFF,' '),OVR)
f) Voraussetzung(en)
Bei PROFILE=ON darf die Option TEST nicht aktiv sein.

3.5 RPTOPTS

a) Syntax
RPTOPTS=((ON/OFF),OVR/NOOVR)
b) Beschreibung
RPTOPTS schreibt Informationen heraus, wie die Optionen gesetzt sind.
c) Performance optimal
RPTOPTS=(OFF),OVR)
d) empfohlen
RPTOPTS=(OFF),OVR)
e) Default
RPTOPTS=(OFF),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Die Option RPTOPTS=ON sollte für einzelne Anwendungen kurzfristig benutzt werden, um die optimalen Optionen festlegen zu können. Dabei ist zu beachten, dass die Performance zu dieser Zeit deutlich schlechter ist.

3.6 RTEREUS

a) Syntax
RTEREUS=((ON/OFF),OVR/NOOVR)
b) Beschreibung
RTEREUS initialisiert implizit die Laufzeitumgebung, damit sie „reusable“ ist, wenn das zugehörende Hauptprogamm im Thread ein COBOL Programm ist.
c) Performance optimal
RTEREUS=((OFF),OVR)
d) empfohlen
RTEREUS=((OFF),OVR)
e) Default
RTEREUS=((OFF),OVR)
f) Voraussetzung(en)
Nur für COBOL Anwendungen, die statt einem STOP RUN einen GOBACK kodiert haben.
Diese Option kann nur in Verbindung mit CEEDOPT, CEEUOPT, CEEROPT oder einem ASM Userexit benutzt werden.
g) Anmerkung(en)
Die Option hat einige Seiteneffekte, die möglicherweise nicht erwünscht sind. Es sollte daher sehr genau überprüft werden, wann diese Option sinnvoll ist. Sie sollte aber stets nur auf der Ebene der Anwendung eingesetzt werden.

3.7 STORAGE

a) Syntax
STORAGE(heap-alloc,heap-free,dsa-alloc,res)
b) Beschreibung
STORAGE(00,00,00) initialisiert den Workbereich des gerufenen Programms mit Low-Value.
c) Performance optimal
STORAGE(NONE,NONE,NONE,8k)
d) empfohlen
STORAGE(NONE,NONE,NONE,8k), wenn NOWSCLEAR benutzt wurde
STORAGE(00,00,NONE,8k), wenn WSCLEAR benutzt wurde
e) Default
STORAGE(NONE,NONE,NONE,0k)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Die Entscheidung, welche der Ausprägung der Option genommen wird, hängt ganz wesentlich davon ab, mit welcher COBOL-Option – NOWSCLEAR oder WSCLEAR – gearbeitet worden ist.
“Never use STORAGE with 3rd parm other than NONE!”

3.8 TERMTHDACT

a) Syntax
TERMTHDACT=((opt,CESE,reg-stor),OVR)
b) Beschreibung
Diese Option wirkt alleine im Abbruchfall und steuert, welche Art von Dump ausgegeben werden soll. CESE gibt den Out-DDNAMEN an, reg-stor den Bereich in Bytes, der rund um die Abbruchadresse gedumpt werden soll.
c) Performance optimal
TERMTHDACT(QUIET)
d) empfohlen
TERMTHDACT(TRACE)
e) Default
TERMTHDACT=((TRACE,CESE,96),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
TERMTHDACT(TRACE) löst bei einem Abbruch den wichtigsten Teil des CEEDUMP einschließlich aller Traceback-Informationen aus. Dieser Dump beinhaltet im Regelfall alle für eine Fehleranalyse benötigten Informationen. Auf diesem Dump können auch Dumpauswertungstools wie Abend-Aid ™ aufsetzen. Bei der Einstellung dieser Option sollte man nicht unbeachtet lassen, was sich in dem Unternehmen bisher „eingebürgert“ hat. Der Performancegewinn mit der optimalen Einstellung ist nicht besonders hoch.

3.9 TEST / NOTEST

a) Syntax
TEST/NOTEST=((ALL/ERROR/NONE,comm-file,command,pref-file), OVR/NOOVR)
b) Beschreibung
TEST gibt die Bedingungen an, wie das DEBUG-TOOL genutzt werden kann.
c) Performance optimal
NOTEST=((ALL,*,PROMPT,INSPPREF),OVR)
d) empfohlen
NOTEST=((ALL,*,PROMPT,INSPPREF),OVR) in Produktionsumgebung
e) Default
NOTEST=((ALL,*,PROMPT,INSPPREF),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Bei Verwendung des Debug-Tools sollte entschieden werden, ob in der Test-Umgebung generell oder nur auf Ebene der Anwendung die Option TEST eingesetzt wird.

3.10 TRAP

a) Syntax
TRAP=((ON/OFF,SPIE/NOPSPIE),OVR/NOOVR)
b) Beschreibung
Diese Option wirkt nur im Abbruchfall und steuert das Condition handling in LE.
c) Performance optimal
TRAP=((ON,SPIE),OVR)
d) empfohlen
TRAP=((ON,SPIE),OVR)
e) Default
TRAP=((ON,SPIE),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
keine

4 non-CICS Optionen - Speicherparameter

4.1 ALL31

a) Syntax
ALL31=((ON/OFF),OVR/NOOVR)

b) Beschreibung
Diese Option legt fest, welche Speicherbereiche LE für die Enklaven nutzen darf. Die Option ändert implizit keine Speicherbereiche, die von den Laufzeitoptionen STACK und HEAP gesetzt werden. Die Option ist für alle (LE-)Sprachen und für einen Sprachenmix zugelassen.

c) Performance optimal
ALL31=((ON),OVR)

d) empfohlen
ALL31=((ON),OVR)

e) Default
ALL31=((ON),OVR)

f) Voraussetzung(en)
1. Für C/C++ und PL/I muss ALL31(OFF) bei AMODE(24) Programmen gesetzt sein.
2. Für COBOL muss ALL31(OFF) gesetzt sein, wenn für die Anwendung eine der folgenden Bedingungen gilt:
   - Es gibt ein VS COBOL II Programm mit NORES.
   - Es gibt ein OS/VS COBOL Programm.
   - Es gibt ein AMODE(24) Programm
3. Zu beachten ist, dass ALL31(OFF) auch STACK(,,BELOW) bedingt.
4. Alle Enklaven müssen die gleiche Einstellung für ALL31 haben.

4.2 ANYHEAP

a) ANYHEAP((init,incr,w1,w2),OVR/NOOVR)
b) Beschreibung
Diese Option legt fest, wie viel Speicher und wo LE diesen Speicher für den Heap-Storage nutzen darf. w1 kann die Werte ANY oder BELOW annehmen, w2 die Werte FREE oder KEEP.
c) Performance optimal
ANYHEAP((16k,8k,ANY,FREE),OVR)
d) empfohlen
ANYHEAP((16k,8k,ANY,FREE),OVR)
e) Default
ANYHEAP=((16K,8K,ANYWHERE,FREE),OVR)
f) Voraussetzung(en)
Falls FORTRAN Programme im Einsatz sind, ist eine andere Parametrisierung erforderlich.
g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

4.3 BELOWHEAP

a) Syntax
BELOWHEAP((init,incr,w1),OVR/NOOVR)
b) Beschreibung
Hiermit wird der HEAP Speicherbereich kontrolliert, der unterhalb der 16 MB Linie liegen muss. Beispiel: Kontrollblöcke für I/O. w1 kann die Werte FREE oder KEEP annehmen.
c) Performance optimal
BELOWHEAP=((8K,4K, FREE),OVR)
d) empfohlen
BELOWHEAP=((8K,4K, FREE),OVR)
e) Default
BELOWHEAP=((8K,4K,FREE),OVR)
f) Voraussetzung(en)
Falls FORTRAN Programme im Einsatz sind, ist eine andere Parametrisierung erforderlich.
g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

4.4 HEAP

a) Syntax
HEAP=((init,incr,ANY/BELOW,KEEP/FREE,init24,incr24),OVR/NOOVR)

b) Beschreibung
HEAP kontrolliert die Bereitstellung und Verwaltung des HEAP-Speichers. Eine Bearbeitung zusätzlicher Bereiche kann über den LE Callable Service CEE-CRHP erfolgen. Heap-Speicher sind Bereiche wie manche C-Variablen, COBOL Working Storage und im PL/I CTR- und BASED-Variablen. LE reserviert den Bereich genau dann, wenn die Anwendung erstmalig solche Bereiche anfordert. *init* gibt das Minimum des bereitgestellten Speicherbereichs an; als Werte sind n, nK oder nM möglich. Es gilt ein Vielfaches von 8 bytes. *incr* gibt das dazugehörige Inkrement an. Die Werte sind wie vor. *init24* und *incr24* sind die entsprechenden Werte für die Bereiche below-the-line.

c) Performance optimal
HEAP=((128K,32K,ANYWHERE,KEEP,8K,4K),OVR)

d) empfohlen
HEAP=((128K,32K,ANYWHERE,KEEP,8K,4K),OVR)

e) Default
HEAP=((32K,32K,ANYWHERE,KEEP,8K,4K),OVR)

f) Voraussetzung(en)
keine.

g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

4.5 HEAPCHK

a) Syntax
HEAPCHK=((OFF/ON,freq,del,cl),OVR/NOOVR)

b) Beschreibung
HEAPCHK=ON kontrolliert den Heap-Speicher. *freq* gibt die Zahl an, bei dem wievielten Aufruf von LE dieser kontrolliert werden soll. *del* gibt die Anzahl nach dem ersten Aufruf von LE an, bei dem zum ersten Mal geprüft werden soll. *cl* ist ein Call-Level für den Traceback Pfad, der zu prüfen ist.

c) Performance optimal
HEAPCHK=((OFF,1,0,0),OVR)

d) empfohlen
HEAPCHK=((OFF,1,0,0),OVR)

e) Default
HEAPCHK=((OFF,1,0,0),OVR)

f) Voraussetzung(en)
keine

g) Anmerkung(en)
Bei HEAPCHK=ON gibt es Auswirkungen auf die Speicherbereiche, die durch die Option STORAGE angesprochen werden.

4.6 HEAPPOOLS

a) Syntax
HEAPPOOLS=((OFF/ON,8,10,32,10,128,10,256,10,1024,10,2048,10),OVR)
b) Beschreibung
HEAPPOOLS kontrolliert einen optionalen zusätzlichen HEAP-Speicherbereich, genannt HEAPPOOL.
c) Performance optimal
HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048,10),OVR)
d) empfohlen
HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048,10),OVR)
e) Default
HEAPPOOLS=((OFF,8,10,32,10,128,10,256,10,1024,10,2048,10),OVR)
f) Voraussetzung(en)
gilt nur für C/C++ und VA PL/I Anwendungen
g) Anmerkungen(en)
keine Detaillierung in diesem Dokument

4.7 LIBSTACK

a) Syntax
LIBSTACK=((init,incr,FREE/KEEP),OVR/NOOVR)
b) Beschreibung
LIBSTACK kontrolliert die Bereitstellung des Stackspeichers für die Thread Bibliothek. Dieser Stack wird von denjenigen LE- und HLL-Routinen genutzt, die Save Areas unter der 16 MB Linie benötigen. *init* gibt das Minimum in n, nK oder nM Bytes des bereitgestellten Speicherbereichs an; der Speicher ist zusammenhängend. *incr* gibt das dazugehörige Inkrement an. Die Werte sind wie vor.
c) Performance optimal
LIBSTACK=((4K,4K,KEEP),OVR)
d) empfohlen
LIBSTACK=((4K,4K,KEEP),OVR)
e) Default
LIBSTACK=((4K,4K,FREE),OVR)
f) Voraussetzung(en)
keine
g) Anmerkungen(en)
Die möglichen Kombinationen von init und incr sind sehr vielfältig. Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

4.8 RPTSTG

a) Syntax
RPTSTG=((ON/OFF),OVR/NOOVR)

b) Beschreibung
RPTSTG gibt nach dem normal beendeten Lauf der Anwendung Informationen, wie die Speicherbereiche genutzt werden.

c) Performance optimal
RPTSTG=(OFF),OVR)

d) empfohlen
RPTSTG=(OFF),OVR)

e) Default
RPTSTG=(OFF),OVR)

f) Voraussetzung(en)
keine

g) Anmerkung(en)
Diese Option sollte für einzelne Anwendungen kurzfristig benutzt werden, um die optimale Speicherzuordnung festlegen zu können. Dabei ist zu beachten, dass die Performance zu dieser Zeit deutlich schlechter ist.
Wichtige Optionen in dieser Hinsicht sind: ANYHEAP, BELOWHEAP, HEAP, LIBSTACK, NONIPTSTACK (neu: THREADSTACK), STACK und THREAD-HEAP. Die aus dem Report erhaltenen Werte (Blocks) sollten für die obigen Optionen als Initialwerte benutzt werden.
Zitat aus CEEA5110: „The storage report generated by RPTSTG(ON) shows the number of system-level get storage calls that were required while the application was running. To improve performance, use the storage report numbers generated by the RPTSTG option as an aid in setting the initial and increment size for STACK, THREADSTACK and HEAP. This reduces the number of times that the Language Environment storage manager makes requests to acquire storage. For example, you can use the storage report numbers to set appropriate values in the HEAP init_size and incr_size fields, and in the STACK and THREADSTACK dsinit_size, dsincr_size, usinit_size and usincr_size fields, for allocating storage.”

4.9 STACK

a) Syntax
STACK((uinit,uincr,BELOW/ANY,KEEP/FREE,dinit,dincr),OVR/NOOVR)
b) Beschreibung
STACK kontrolliert den Speicherbereich, der den Stack auf- und wieder abbaut. Typische Felder, die dort abgelegt werden, sind automatic Variablen in C und PL/I und LOCAL-STORAGE in COBOL, sowie Work-Bereiche für COBOL-Bibliotheksroutinen. Nicht abgelegt werden z.B. CAA und andere Kontrollblöcke. Die Werte *dinit* und *dincr* gelten nur für Anwendungen, die durch XPLINK gerufen werden.
c) Performance optimal
STACK(128K,128K,ANYWHERE,KEEP,512K,128K)
d) empfohlen
STACK(64K,64K,BELOW,KEEP,512K,128K) für reine COBOL-Anwendungen aber
STACK(128K,128K,ANYWHERE ,KEEP,512K,128K) für reine 31-bit-Anwendungen und nur wenn ALL31 gesetzt ist
e) Default
STACK(128K,128K, ANYWHERE,KEEP,512K,128K)
f) Voraussetzung(en)
STACK(,,BELOW,,) muss gesetzt sein, wenn ALL31(OFF) gilt.
g) Anmerkung(en)
keine

4.10 THREADHEAP

a) Syntax
THREADHEAP((init,incr,BELOW/ANY,KEEP/FREE),OVR/NOOVR)
b) Beschreibung
THREADHEAP kontrolliert den THREAD-LEVEL-HEAP-Speicherbereich.
c) Performance optimal
THREADHEAP=((8K,8K,ANYWHERE,KEEP),OVR)
d) empfohlen
THREADHEAP=((8K,8K,ANYWHERE,KEEP),OVR)
e) Default
THREADHEAP=((4K,4K,ANYWHERE,KEEP),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

4.11 THREADSTACK

a) Syntax
THREADSTACK= ((OFF/ON,uinit,uincr,BELOW/ANY,KEEP/FREE,dinit,dincr), OVR/NOOVR)

b) Beschreibung
THREADHEP kontrolliert den THREAD-Speicherbereich in einer Multi-THREAD-Anwendung mit Ausnahme des Initial-Thread-Bereichs.

c) Performance optimal
THREADSTACK=((OFF,8K,8K,ANYWHERE,KEEP,128K,128K),OVR)

d) empfohlen
THREADSTACK=((OFF,8K,8K,ANYWHERE,KEEP,128K,128K),OVR)

e) Default
THREADSTACK=((OFF,4K,4K,ANYWHERE,KEEP,128K,128K),OVR)

f) Voraussetzung(en)
keine

g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

5 CICS Optionen - ohne Speicherparameter

5.1 CBLPSHPOP

a) Syntax
CBLPSHPOP=((ON/OFF),OVR/NOOVR)
b) Beschreibung
Diese Option gibt an, wie der COBOL-Call zu behandeln ist. Ist die Option auf ON gesetzt, wird beim Aufruf des Unterprogramms ein PUSH HANDLE abgesetzt mit allen Konsequenzen.
c) Performance optimal
CBLPSHPOP=((OFF),OVR)
d) empfohlen
CBLPSHPOP=((OFF),OVR)
CBLPSHPOP=((ON),OVR) bei einem Mix von COBOL-Versionen untereinander.
e) Default
CBLPSHPOP =((ON),OVR)
f) Voraussetzung(en)
Die Option gilt nur für COBOL-Programme.

5.2 CHECK

a) Syntax
CHECK=((ON/OFF),OVR/NOOVR)
b) Beschreibung
Während der Ausführungszeit werden Fehler in einer COBOL-Anwendung geflaggt, die sich auf Indices, Subscripte und Referenzmodifikationen beziehen.
c) Performance optimal
CHECK=((OFF),OVR)
d) empfohlen
CHECK=((ON),OVR)
e) Default
CHECK=((ON),OVR)
f) Voraussetzung(en)
COBOL-Anwendung
g) Anmerkung(en)
Nach der IBM-Literatur hat die Option CHECK=ON keinerlei Auswirkung auf die Performance, wenn die Anwendung mit NOSSRANGE umgewandelt worden ist. Untersuchungen haben zwar das Gegenteil gezeigt, trotzdem wird CHECK=ON empfohlen. Gleichzeitig sollte aber die COBOL-Compile-Option NOSSRANGE aktiviert sein. In Ausnahmefällen darf SSRANGE kurze Zeit benutzt werden.

5.3 DEBUG

a) Syntax
DEBUG / NODEBUG bzw. DEBUG=((ON/OFF),OVR/NOOVR)
b) Beschreibung
DEBUG aktiviert die COBOL Batch DEBUG Facility, die in den Declaratives (USE FOR DEBUGGING) gesetzt worden ist.
c) Performance optimal
NODEBUG
d) empfohlen
NODEBUG für Produktion, DEBUG für Test
e) Default
NODEBUG
f) Voraussetzung(en)
COBOL-Anwendung

5.4 PROFILE

a) Syntax
PROFILE=((ON/OFF,' '),OVR/NOOVR)
b) Beschreibung
PROFILE kontrolliert einen Profiler, der Performance Daten der Anwendung sammelt.
c) Performance optimal
PROFILE=((OFF,' '),OVR)
d) empfohlen
PROFILE=((OFF,' '),OVR)
e) Default
PROFILE=((OFF,' '),OVR)
f) Voraussetzung(en)
Bei PROFILE=ON darf die Option TEST nicht aktiv sein.

5.5 RPTOPTS

a) Syntax
RPTOPTS=((ON/OFF),OVR/NOOVR)
b) Beschreibung
RPTOPTS gibt nach dem Lauf der Anwendung Informationen, wie die Optionen gesetzt sind.
c) Performance optimal
RPTOPTS=(OFF),OVR)
d) empfohlen
RPTOPTS=(OFF),OVR)
e) Default
RPTOPTS=(OFF),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Diese Option sollte für einzelne Anwendungen kurzfristig benutzt werden, um die optimalen Optionen festlegen zu können. Dabei ist zu beachten, dass die Performance zu dieser Zeit deutlich schlechter ist.

5.6 STORAGE

a) Syntax
STORAGE(heap-alloc,heap-free,dsa-alloc,res)
b) Beschreibung
STORAGE(00,00,00) initialisiert den Workbereich des gerufenen Programms mit Low-Value.
c) Performance optimal
STORAGE(NONE,NONE,NONE,8k)
d) empfohlen
STORAGE(NONE,NONE,NONE,8k), wenn NOWSCLEAR benutzt wurde
STORAGE(00,00,NONE,8k), wenn WSCLEAR benutzt wurde
e) Default
STORAGE(NONE,NONE,NONE,0k)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Die Entscheidung, welche der Ausprägung der Option genommen wird, hängt ganz wesentlich davon ab, mit welcher COBOL-Option – NOWSCLEAR oder WSCLEAR – gearbeitet worden ist.
“Never use STORAGE with 3rd parm other than NONE!”

5.7 TERMTHDACT

a) Syntax
TERMTHDACT=((opt,CESE,reg-stor),OVR/NOOVR)
b) Beschreibung
Diese Option wirkt alleine im Abbruchfall und steuert, welche Art von Dump ausgegeben werden soll. CESE gibt den Out-DDNAMEN an, reg-stor den Bereich, der rund um die Abbruchadresse gedumpt werden soll.
c) Performance optimal
TERMTHDACT(QUIET)
d) empfohlen
TERMTHDACT(TRACE)
e) Default
TERMTHDACT=((TRACE,CESE,96),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
TERMTHDACT(TRACE) löst bei einem Abbruch den wichtigsten Teil des CEEDUMP einschließlich aller Traceback-Informationen aus. Dieser Dump beinhaltet im Regelfall alle für eine Fehleranalyse benötigten Informationen. Auf diesem Dump können auch Dumpauswertungstools wie Abend-Aid ™ aufsetzen. Bei der Einstellung dieser Option sollte man nicht unbeachtet lassen, was sich in dem Unternehmen bisher „eingebürgert“ hat. Der Performancegewinn mit der optimalen Einstellung ist nicht besonders hoch.

5.8 TEST / NOTEST

a) Syntax
TEST/NOTEST=((ALL/ERROR/NONE,comm-file,command,pref-file), OVR/NOOVR)
b) Beschreibung
TEST gibt die Bedingungen an, wie das DEBUG-TOOL genutzt werden kann.
c) Performance optimal
NOTEST=((ALL,*,PROMPT,INSPPREF),OVR)
d) empfohlen
NOTEST=((ALL,*,PROMPT,INSPPREF),OVR) in Produktionsumgebung
e) Default
NOTEST=((ALL,*,PROMPT,INSPPREF),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Bei Verwendung des Debug-Tools sollte entschieden werden, ob in der Test-Umgebung generell oder nur für eine Anwendung die Option TEST eingesetzt wird.

5.9 TRAP

a) Syntax
TRAP=((ON/OFF,SPIE/NOPSPIE),OVR/NOOVR)
b) Beschreibung
Diese Option wirkt nur im Abbruchfall und steuert das Condition handling in LE.
c) Performance optimal
TRAP=((ON,NOSPIE),OVR)
d) empfohlen
TRAP=((ON,NOSPIE),OVR)
e) Default
TRAP=((ON,SPIE),OVR)
f) Voraussetzung(en)
keine
g) Anmerkung(en)
Da LE unter CICS niemals einen SPIE oder STAE setzt, wird die SPIE/NOSPIE Option unter CICS ignoriert.

6 CICS Optionen - Speicherparameter

6.1 ALL31

a) Syntax
ALL31=((ON/OFF),OVR/NOOVR)

b) Beschreibung
Diese Option legt fest, welche Speicherbereiche LE für die Enklaven nutzen darf. Die Option ändert implizit keine Speicherbereiche, die von den Laufzeitoptionen STACK und HEAP gesetzt werden.

c) Performance optimal
ALL31=((ON),OVR)

d) empfohlen
ALL31=((ON),OVR)

e) Default
ALL31=((ON),OVR)

f) Voraussetzung(en)
1. Für C/C++ und PL/I muss ALL31(OFF) bei AMODE(24) Programmen gesetzt sein.
2. Für COBOL muss ALL31(OFF) gesetzt sein, wenn für die Anwendung eine der folgenden Bedingungen gilt:
   - Es gibt ein VS COBOL II Programm mit NORES.
   - Es gibt ein OS/VS COBOL Programm.
   - Es gibt ein AMODE(24) Programm
3. Zu beachten ist, dass ALL31(OFF) auch STACK(,,BELOW) bedingt.
4. Alle Enklaven müssen die gleiche Einstellung für ALL31 haben.

g) Bemerkung(en)
Die Einstellung ALL31=OFF sollte möglichst vermieden werden. Sollte es unbedingt nötig sein, sollte man diese Anwendungen separieren.

6.2 ANYHEAP

a) ANYHEAP((init,incr,w1,w2),OVR/NOOVR)
b) Beschreibung
Diese Option legt fest, wie viel Speicher und wo LE diesen Speicher für den Heap-Storage nutzen darf. w1 kann die Werte ANY oder BELOW annehmen, w2 die Werte FREE oder KEEP.
c) Performance optimal
ANYHEAP((16k,4080,ANY,FREE),OVR)
d) empfohlen
ANYHEAP((16k,4080,ANY,FREE),OVR)
e) Default
ANYHEAP=((4K,4080,ANYWHERE,FREE),OVR)
f) Voraussetzung(en)
Falls FORTRAN Programme im Einsatz sind, ist eine andere Parametrisierung erforderlich.
g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

6.3 BELOWHEAP

a) Syntax
BELOWHEAP((init,incr,w1),OVR/NOOVR)
b) Beschreibung
Hiermit wird der HEAP Speicherbereich kontrolliert, der unterhalb der 16 MB Linie liegen muss. Beispiel: Kontrollblöcke für I/O. w1 kann die Werte FREE oder KEEP annehmen.
c) Performance optimal
BELOWHEAP=((8K,4080, FREE),OVR)
d) empfohlen
BELOWHEAP=((8K,4080, FREE),OVR)
e) Default
BELOWHEAP=((4K,4080,FREE),OVR)
f) Voraussetzung(en)
Falls FORTRAN Programme im Einsatz sind, ist eine andere Parametrisierung erforderlich.
g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

6.4 HEAP

a) Syntax
HEAP=((init,incr,ANY/BELOW,KEEP/FREE,init24,incr24),OVR/NOOVR)

b) Beschreibung
HEAP kontrolliert die Bereitstellung und Verwaltung des HEAP-Speichers. Eine Bearbeitung zusätzlicher Bereiche kann über den LE Callable Service CEE-CRHP erfolgen. Heap-Speicher sind Bereiche wie manche C-Variablen, COBOL Working Storage und im PL/I CTR- und BASED-Variablen. LE reserviert den Bereich genau dann, wenn die Anwendung erstmalig solche Bereiche anfordert. *init* gibt das Minimum des bereitgestellten Speicherbereichs an; als Werte sind n, nK oder nM möglich. Es gilt ein Vielfaches von 8 bytes. *incr* gibt das dazugehörige Inkrement an. Die Werte sind wie vor. *init24* und *incr24* sind die entsprechenden Werte für die Bereiche below-the-line.

c) Performance optimal
HEAP=((32K,4080,ANYWHERE,KEEP,8K,4080),OVR)

d) empfohlen
HEAP=((32K,4080,ANYWHERE,KEEP,8K,4080),OVR)

e) Default
HEAP=((4k,4080,ANYWHERE,KEEP,4K,4080),OVR)

f) Voraussetzung(en)
keine.

g) Anmerkung(en)
Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

6.5 HEAPCHK

a) Syntax
HEAPCHK=((OFF/ON,freq,del,cl),OVR/NOOVR)

b) Beschreibung
HEAPCHK=ON kontrolliert den Heap-Speicher. *freq* gibt die Zahl an, bei dem wievielten Aufruf von LE dieser kontrolliert werden soll. *del* gibt die Anzahl nach dem ersten Aufruf von LE an, bei dem zum ersten Mal geprüft werden soll. *cl* ist ein Call-Level für den Traceback Pfad, der zu prüfen ist.

c) Performance optimal
HEAPCHK=((OFF,1,0,0),OVR)

d) empfohlen
HEAPCHK=((OFF,1,0,0),OVR)

e) Default
HEAPCHK=((OFF,1,0,0),OVR)

f) Voraussetzung(en)
keine

g) Anmerkung(en)
Bei HEAPCHK=ON gibt es Auswirkungen auf die Speicherbereiche, die durch die Option STORAGE angesprochen werden.

6.6 HEAPPOOLS

a) Syntax
HEAPPOOLS=((OFF/ON,8,10,32,10,128,10,256,10,1024,10,2048,10),OVR)
b) Beschreibung
HEAPPOOLS kontrolliert einen optionalen zusätzlichen HEAP-Speicherbereich.
c) Performance optimal
HEAPPOOLS=((OFF,8,10,32,10,128,256,10,1024,10,2048,10),OVR)
d) empfohlen
HEAPPOOLS=((OFF,8,10,32,10,128,256,10,1024,10,2048,10),OVR)
e) Default
HEAPPOOLS=((OFF,8,10,32,10,128,256,10,1024,10,2048,10),OVR)
f) Voraussetzung(en)
gilt nur für C/C++ und VA PL/I Anwendungen
g) Anmerkungen(en)
keine Detaillierung in diesem Dokument

6.7 LIBSTACK

a) Syntax
LIBSTACK=((init,incr,FREE/KEEP),OVR/NOOVR)
b) Beschreibung
LIBSTACK kontrolliert die Bereitstellung des Stackspeichers für die Thread Bibliothek. Dieser Stack wird von denjenigen LE- und HLL-Routinen genutzt, die Save Areas unter der 16 MB Linie benötigen. *init* gibt das Minimum in n, nK oder nM Bytes des bereitgestellten Speicherbereichs an; der Speicher ist zusammenhängend. *incr* gibt das dazugehörige Inkrement an. Die Werte sind wie vor.
c) Performance optimal
LIBSTACK=((4K,4080,KEEP),OVR)
d) empfohlen
LIBSTACK=((4K,4080,FREE),OVR)
e) Default
LIBSTACK=((32,4080,FREE),OVR)
f) Voraussetzung(en)
keine
g) Anmerkungen(en)
Die möglichen Kombinationen von init und incr sind sehr vielfältig. Diese Option kann bei Bedarf pro Anwendung verschieden gesetzt werden. Mit der Runoption RPTSTG kann der Speicherbereich, den die Anwendung nutzt, angezeigt werden.

6.8 RPTSTG

a) Syntax
RPTSTG=((ON/OFF),OVR/NOOVR)

b) Beschreibung
RPTSTG gibt nach dem normal beendeten Lauf der Anwendung Informationen, wie die Speicherbereiche genutzt werden.

c) Performance optimal
RPTSTG=(OFF),OVR)

d) empfohlen
RPTSTG=(OFF),OVR)

e) Default
RPTSTG=(OFF),OVR)

f) Voraussetzung(en)
keine

g) Anmerkung(en)
Diese Option sollte für einzelne Anwendungen kurzfristig benutzt werden, um die optimale Speicherzuordnung festlegen zu können. Dabei ist zu beachten, dass die Performance zu dieser Zeit deutlich schlechter ist. Wichtige Optionen in dieser Hinsicht sind: ANYHEAP, BELOWHEAP, HEAP, LIBSTACK, NONIPTSTACK (neu: THREADSTACK), STACK und THREAD-HEAP. Die aus dem Report erhaltenen Werte (Blocks) sollten für die obigen Optionen als Initialwerte benutzt werden.
Zitat aus CEEA5110: „The storage report generated by RPTSTG(ON) shows the number of system-level get storage calls that were required while the application was running. To improve performance, use the storage report numbers generated by the RPTSTG option as an aid in setting the initial and increment size for STACK, THREADSTACK and HEAP. This reduces the number of times that the Language Environment storage manager makes requests to acquire storage. For example, you can use the storage report numbers to set appropriate values in the HEAP init_size and incr_size fields, and in the STACK and THREADSTACK dsinit_size, dsincr_size, usinit_size and usincr_size fields, for allocating storage.”

6.9 STACK

a) Syntax
STACK((uinit,uincr,BELOW/ANY,KEEP/FREE,dinit,dincr),OVR/NOOVR)
b) Beschreibung
STACK kontrolliert den Speicherbereich, der den Stack auf- und wieder abbaut. Typische Felder, die dort abgelegt werden, sind automatic Variablen in C und PL/I und LOCAL-SORAGE in COBOL, sowie Work-Bereiche für COBOL-Bibliotheksroutinen. Nicht abgelegt werden z.B. CAA und andere Kontrollblöcke. Die Werte dinit und dincr sind nur für Anwendungen, die durch XPLINK gerufen werden und werden unter CICS ignoriert.
c) Performance optimal
STACK=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR).
d) empfohlen
STACK=((4K,4080,BELOW,KEEP,4K,4080),OVR).für reine COBOL-Anwendungen aber
STACK=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR).für reine 31-bit-Anwendungen und nur wenn ALL31 gesetzt ist
e) Default
STACK=((4K,4080,ANYWHERE,KEEP,4K,4080),OVR).
f) Voraussetzung(en)
STACK(,,BELOW,,) muss gesetzt sein, wenn ALL31(OFF) gilt.
g) Anmerkung(en)
keine

6.10 THREADHEAP

Diese Option wird unter CICS ignoriert.

6.11 THREADSTACK

Diese Option wird unter CICS ignoriert.

7 Schlussbemerkungen

7.1 Allgemeine Hinweise

- Reentrant LE Module könnten in die LPA gelegt werden, bei CICS in die EPLA.
- SCEELPA kann in die LPALST gelegt werden.
- SCEERUN sollte in der LNKLSR enthalten sein.
- Die Performance wird deutlich besser, wenn die System-Routinen, die Speicheroperationen durchführen, möglichst wenig aufgerufen werden. Dies gilt für HEAP- und STACK-Speicher.
- Sollte noch ein VS COBOL II Compiler im Einsatz sein, sollte dieser schnellstens abgelöst werden. Die Calls sind sehr viel schneller unter dem „neuen“ COBOL.
- STACK ist besonders wichtig zu beobachten, wenn COBOL interne Programme aufruft.
- Unter CICS und mit “alten” LE-Leveln sollte PQ14888 für Performanceverbesserungen eingespielt worden sein.
- Unter CICS sollten die Parameter für das Nachladen von Speicherbereich 16 Bytes weniger als die nk-Grenze betragen, um die Check Zone zu berücksichtigen. Es ist zu wählen 4080 statt 4K oder 1008 statt 1k.

7.2 Zusammenfassung

Es ist unmöglich, in einem bewusst kurz gehaltenen Dokument alle Eventualitäten zu berücksichtigen, die erforderlich sind, um die Optionen so zu wählen, dass alle Anwendungen performant arbeiten. Dieses Dokument kann nur ein erster und einzelner Schritt am Anfang des Optimierungsprozesses sein. Die Umsetzung der Optionen auf die im Dokument empfohlenen ist teilweise nicht möglich, ohne Vorarbeiten gemacht zu haben. Daher schließt dieses Dokument im folgenden Kapitel mit einer Empfehlung, wie die Optionen sinnvoll mittelfristig optimiert werden können.

8 Empfehlungen für die weitere Vorgehensweise

8.1 Basis der Empfehlungen

Basis der Empfehlungen sind die in dem Dokument „Die Optionen von Language Environment unter dem Gesichtspunkt der Performance“ getroffenen Äußerungen und die den Autoren zur Verfügung gestellten Unterlagen:

- TSO-Batch Optionen: Options Report for Enclave XADBUG 11/23/01 9:26:53 AM Language Environment V02 R10.00
- CICP5-Optionen: ohne Überschrift und Datum (im Original auf der Rückseite des vorher genannten Dokuments, daher vermutlich vom gleichen Tag)
- Auswertung COBOL-Optionen und –Befehle mit Optimierungspotential vom 22.11.2001.

Alle Dokumente beziehen sich auf die Umgebung bei

8.2 Erste Optimierungsschritte

Die folgenden Schritte können und sollten schnell umgesetzt werden. Die Änderungen können sofort, ohne Test durchgeführt werden:

8.2.1 Änderungen der Optionen von LE

1. Bei den Speicherparametern im CICS sind die incr-Suboptionen auf xk-Grenze minus 16 zu bringen, d.h. statt 4k: 4080 statt 1k: 1008.

2. Die Option DEBUG ist in Produktion auf NODEBUG bzw. DEBUG=OFF zu setzen.

3. In der Option STORAGE kann der 3. Parameter, der den Initialwert der DSA angibt, auf NONE gesetzt werden. Dieser Bereich wird stets von LE beschrieben, so dass ein Initialwert überflüssig ist.

4. Die Option TERMTHDACT kann optimiert werden; doch ist es sinnvoll, eine eingespielte Logik nicht zu durchbrechen. Der Gewinn bei Umstellung dieser Option ist bezüglich der Performance relativ gering, bezüglich des verbrauchten Speicherplatzes kann er allerdings sehr hoch sein.

5. Bei Verwendung des Debug-Tools der Firma IBM sollte überlegt werden, wie die Option TEST eingesetzt werden soll. Wir empfehlen den Einsatz der Option auf der Ebene der Anwendung. Dies hat unter Umständen einen höheren Aufwand in der Anwendungsentwicklung oder im Bereich der Methoden zur Folge.

8.2.2 Programmoptimierung

1. Programme mit der Umwandlungsoption NOOPTIMIZE können ohne Änderung mit OPTIMIZE umgewandelt werden.

2. Programme mit der Umwandlungsoption SSRANGE können ohne Änderung mit NOSSRANGE umgewandelt werden.

3. Programme, die viel I/O-Geschäft auf QSAM-Dateien betreiben, sollten mit der Umwandlungsoption FASTSRT umgewandelt werden.

4. Programme, die I/O-Geschäft betreiben, sollten ebenfalls die Umwandlungsoption AWO statt NOAWO enthalten.

5. Programme mit der Umwandlungsoption NUMPROC(MIG) brauchen sehr viel mehr Zeit als Programme mit NUMPROC(PFD).

6. Programme mit der Umwandlungsoption TRUNC(BIN) sollten mit der Option TRUNC(STD) umgewandelt werden. Ist es sichergestellt, dass über die nächsten Jahre hinaus die Firma IBM den Compiler stellt, dann kann auch auf die Option TRUNC(OPT) umgestellt werden. Diese Option gehört nicht zum ANSI-Standard, sondern ist eine IBM-Extension.

7. Programme, die mit VS COBOL II (R4) umgewandelt worden sind, können ohne irgend eine Änderung neu mit dem neuesten COBOL-Compiler übersetzt werden. Der CALL ist dann erheblich schneller.

8. Alle COBOL-Programme sind mit AMODE(31) und RMODE(ANY) neu zu linken. Eine Änderung des Codes ist nicht erforderlich.

9. In den COBOL-Programmen ist bei Nutzung von QSAM-Files die Angabe BLOCK CONTAINS 0 RECORDS anzugeben. Gleichzeitig ist in der JCL die BLKSIZE-Angabe zu entfernen.

8.2.3 Sonstige Maßnahmen

1. Alle Defaultwerte für die Compile- und Link-Optionen sind rechtzeitig vor der ersten Aktion auf die optimalen Werte einzustellen.

8.3 Optimierungen mit Vorarbeiten

Das wesentliche Ziel des Prozesses ist eine optimale Ausnutzung der Speicherbereiche. Ins Blickfeld gerät die 31-bit-Adressierung. Im Hinblick auf die Erfordernis der Firmen, immer größere Datenmengen in immer schnellerer Zeit anbieten zu müssen, müssen Speicherbereiche schnell adressierbar und abrufbar sein. Der Weg geht hierbei in Richtung 64-bit-Adressierung. Auch auf dem Großrechner wird es nicht mehr lange dauern, bis diese Architektur angeboten wird. Daher müssen die Firmen schnell die Vorkehrungen treffen, diese Adressierung bald ausnutzen zu können.
Einer der ersten Schritte ist, alle Großrechner-Anwendungen auf die 31-bit-Adressierung zu heben. Dazu dürfen nicht die neuesten Compiler „einfach so“ eingesetzt oder Compile-Optionen und Laufzeitoptionen von LE „hochgesetzt“ werden. Es bedarf einer konzentrierten Projektvorgehensweise, die nachfolgend kurz beschrieben wird.

8.3.1 ASM-Module

Die vorhandenen ASM-Module sind auf AMODE(31) und RMODE(ANY) umzustellen. Dies ist nicht ganz so einfach wie in der Sprache COBOL, denn das Coding selbst muss darauf eingestellt sein. Dass die Registerbehandlung LE-konform ist, wird dabei vorausgesetzt.

8.3.2 Module in „alten“ COBOL-Sprachen

Module in OS/VS COBOL sind auf das neueste COBOL-Release umzustellen. Die Migrationsstrategie, die in verschiedenen IBM-Broschüren beschrieben ist, sollte beachtet werden. Aus der bisherigen Erfahrung heraus, gibt es sehr wenig Probleme bei dieser Migration; fast alle Programme können im Code unverändert übernommen werden. Module in VS COBOL II R4 sollten nach der Umsetzung der Empfehlungen aus dem vorigen Kapitel nicht mehr vorhanden sein.

8.3.3 volle 31-bit-Adressierung in COBOL

Die COBOL-Module sind auf DATA(31) umzustellen. Wenn dies nicht in einer großen Aktion gemacht werden kann, muss dies schrittweise pro Anwendung in der CALL Kette von unten nach oben erfolgen. Vorausgesetzt wird, dass alle ASM-Module bereits umgestellt worden sind. Zu beachten ist ferner, dass Anwendungen mit einem Mix von Sprachen diesen Punkt sehr vorsichtig angehen sollten. Typische aber äußerst seltene Fehler sind, dass irgendwo in einer früheren CALL Kette ein Modul vorhanden war, welches nicht 31-bit-fähig war. Abbruchcodes 0C1 und 0C4 können dann auftreten.

8.3.4 volle 31-bit-Adressierung in LE

Sobald alle Module auf 31-bit-Adressierung umgestellt sind, können die LE-Optionen auf 31-bit bzw. ANYWHERE gehoben werden. Betroffen hiervon sind alle Optionen, die in den entsprechenden Kapiteln dieser Ausarbeitung aufgeführt sind. Dies kann in einem Zug ohne Tests erfolgen.

8.3.5 Separieren von schwierigen Anwendungen

Falls es Anwendungen oder Teile von Anwendungen gibt, die nicht 31-bit fähig sind, werden diese derart separiert, dass dafür suboptimale LE-Options auf der Ebene der Anwendung definiert werden. Der Begriff Anwendung ist im Sinne von LE zu verstehen. Alle Module innerhalb einer Anwendung, auch wenn diese über mehrere Enklaven geht, müssen 31-bit-adressierbar sein, damit die LE-Optionen auf 31-bit angehoben werden können.

8.3.6 weiteres Optimieren der LE Optionen

Anschließend werden Performance-kritische Anwendungen mit RPTSTG=ON detailliert untersucht, damit für diese optimierte LE-Optionen eingestellt werden können. Das Vorgehen ist wie folgt: Die zu untersuchende Anwendung wird mit der Option RPTSTG=ON gestartet. Der Report gibt für alle speicherrelevanten Optionen die gefundenen Werte aus. An Hand dieser Werte sollten die Einstellungen der Optionen so optimiert werden, dass die Anzahl der Zugriffe zur Bereitstellung von Speicher möglichst gering ist. Danach ist die Anwendung erneut zu starten; das Ergebnis kann mit der Option RPTSTG=ON kontrolliert und bei Bedarf erneut angepasst werden. Da die Anwendungen verändert werden, ist dies ein Prozess, der immer wieder durchlaufen werden sollte. Ein Ziel dabei könnte sein, dass die allgemeinen Optionen von LE so eingestellt werden, dass auch die Performance-kritischen Anwendungen mit diesen Optionen laufen können. Dies ist unter der 31-bit Adressierung vielleicht noch nicht vollständig umsetzbar, bei einer 64-bit-Adressierung wird dies aber machbar werden.

8.3.7 COBOL Programmierung

Gleichzeitig zu den gesamten Aktionen sollten die Programme auf CPU-intensiven Code untersucht werden. Hierbei sind vor allem die Befehle INITIALIZE, INSPECT, STRING und UNSTRING interessant. Es geht nicht darum, diese Befehle aus den Programmen zu eliminieren, sondern darum, ehrlich zu hinterfragen, ob es keine Alternative gibt, die weniger Ressourcen benötigt.

8.3.8 Das Datenmodell und die Implementierung in DB2

Es sollte nicht vergessen werden, dass das Datenmodell der Fachanwendungen, das als Tables im DB2 abgelegt ist, den größten Einfluss auf die Performance hat. Ein Datenmodell, das gründlich durchdacht genau die Anforderungen abdeckt, ist das Beste, was einem System passieren kann. Jede technische Optimierung ist in großen Teilen umsonst, wenn die Datenkonstellation performantes Arbeiten nicht zulässt.

9 Quellennachweis

- Customizing OS/390 Language Environment, Mary Astley, SHARE, Session 2831, July 27, 2000
- Performance Tips and Techniques for COBOL and Language Environment, Tom Ross, SHARE Session: 8214, March, 2000
- LE Stack & Heap processing, John Monti, SHARE Sessions 8209/8210, Jul 2001
- Performance Tips and Techniques for COBOL, PL/ I and Language Environment, Tom Ross, SHARE Session: 8213, July, 2001
- Latest on CICS Applications and LE, Andy Krasun, SHARE Session 8235, Jul 2001
- LE Options for IMS Systems, Terry Seibert, SHARE Session 1233/ 1234, 8150/ 8151 & 8284/ 8285, July 25, 2001
- Language Environment Setup & Customization, Jerry Moody, SHARE Session 8207, February 2001
- Migrating CICS COBOL &PL/I to LE, Tom Ross and Don Smith, SHARE Session 8217, July 2000
- Diagnosing Application Problems Under LE, Mark Picard, SHARE Session 8208, Feb 2001
- IBM Broschüre SA22-7564-01 Language Environment Customization
- IBM Broschüre SA22-7561-01 z/OS V1R2.0 Language Environment Programming Guide
- IBM Broschüre SA22-7562-01 z/OS V1R2.0 Language Environment Programming Reference
- IBM Broschüre SA22-7566-01 z/OS V1R2.0 Language Environment Run-Time Messages
- IBM Broschüre GA22-7565-01 z/OS V1R2.0 Language Environment Run-Time Migration Guide
- IBM Broschüre SA22-7567-01 z/OS V1R2.0 Language Environment Concepts Guide
- IBM Broschüre GC28-1945-09 OS/390 V2R10.0 Language Environment for OS/390 & VM Concepts Guide

Alle vor genannten Informationen sind zu finden unter:
http://www-1.ibm.com/servers/eserver/zseries/zos/le/library/library.html

10 Änderungen gegenüber der vorherigen Version

- 8.3.5 Separieren von schwierigen Anwendungen