

RACF & Performance



GSE Regional Conference – Halle – 19/11/2008

November 26, 2008

Jan De Decker, Fortis

Agenda

- RACF in action
- RACF Processing
- Tuning options
- Recommendations

RACF in action: Collecting data

■ SAFTRACE

- RACF Subsystem must be running

- GTF must be started with (parmlib member or reply:

- TRACE=USRP
 - USR=(F44),END

- SAF trace is activated with operator command:

SAF Trace

- Stop GTF
- STOP The SAF trace
- CHECK WITH RACF SET LIST:

```
TRACE OPTIONS          - NOIMAGE
                      - NOAPPC
                      - NOSYSTEMSSL
                      - NORACROUTE
                      - NOCALLABLE
                      - NOPDCALLABLE
                      - NODATABASE
                      - NOASID
                      - NOJOBNAME
```

SAF Trace Analysis

- IPCS 0 (set the name of the GTF trace)
- 2 Analysis
- 7 Traces
- 2 GTF Trace
- Start with s USR

SAF Trace analysis

- Or use IPCS command:
 - IPGTF USR
- Or write a program yourself

SAF Trace Analysis

- Raw format looks like:
-KC.ÿ..ê.....GTS .. .SP7.0.8 HBB7730 SYC1
- .iC.ÿ.'ó&TÕà.4á.ZRDLRCIRTRACE RACFPRE
- .iC.ÿ.'JútÕà.4á.ZRDLRCIRTRACE RACFPOST.....
- .iC.ÿ.=ođpÕà.³L.RACFRTRACE RACFPRE
- .iC.ÿ.=æ.pÕà.³L.RACFRTRACE RACFPOST.....
- .iC.ÿ.=αG.Õà.4á.ZRDLRCIRTRACE RACFPRE
- .iC.ÿ.=sldÕà.4á.ZRDLRCIRTRACE RACFPOST.....
- .iC.ÿ.=·w.Õà.0¥ØDAMPSFRTRACE RACFPRE
- .iC.ÿ.=¼.-Õà.0¥ØDAMPSFRTRACE RACFPOST.....
- .iC.ÿ.=Jú Õà.7.ØG36545ARTRACE RACFPRE
- .iC.ÿ.=KZtÕà.4á.ZRDLRCIRTRACE RACFPRE
- .iC.ÿ.=Mq.Õà.7.ØG36545ARTRACE RACFPOST.....
- .iC.ÿ.=ÿª.Õà.4á.ZRDLRCIRTRACE RACFPOST.....

RACTRACE

- Available as assembler source from the IBM site
- Basically the RACF postprocessing exits and 2 TSO commands (RACTRACE & RACTR):
 - ICHRCX02
 - ICHRDY02
 - ICHRFY02
 - ICHRFY04
 - ICHRIX02
 - ICHRLX01
 - ICHRTXU0
 - ICHRTXV0
 - ICHRTX00
- Usage is protected by FACILITY class profiles

RACTRACE Output Example: SDSF (1)

- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFISP PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFPARMS PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFSTAE PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFIO PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFTINIT PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFUSER PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFVTBL PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFS4DSP PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 FASTAUTH REQSTOR=FAE SUBSYS=CONTENTS
- RACF ROUTER2 CLASS=PROGRAM MODULE=ISFMMSGH PREVIOUS MODULE UNCONTROLLED
- RACF *ROUTER1 AUTH REQSTOR=ISFGROUP SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF GROUP.ISFUSER.SDSF
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**

RACTRACE Output Example: SDSF (2)

- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF GROUP.ISFUSER.SDSF
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NONE CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFIPREF SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.PREFIX
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.PREFIX
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFIOWNR SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.OWNER
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.OWNER
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFIFLIM SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.FINDLIM

RACTRACE Output Example: SDSF (3)

- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.FINDLIM
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFISYSI SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.SYSID
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.SYSID
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFISYSN SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.SYSNAME
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.SYSNAME
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFIRSYS SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF

RACTRACE Output Example: SDSF (4)

- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.RSYS
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.RSYS
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFIACTN SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.ACTION
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.ACTION
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00
- RACF *ROUTER1 AUTH REQSTOR=ISFINPUT SUBSYS=**NONE** DECOUPL=YES
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE** DECOUPL=YES
- RACF ROUTER2 CLASS=SDSF
- RACF ROUTER2 CLASS=SDSF ISFCMD.FILTER.INPUT
- RACF *ROUTER1 STAT REQSTOR=**NONE** SUBSYS=**NONE**
- System SYT2 shutdown in progress. Please save your work and logoff. CN(SYT1SS04)
- RACF ROUTER2 CLASS=SDSF
- RACF RACHECK1 READ SDSF ISFCMD.FILTER.INPUT
- RACF RACHECK2 DSTYPE=N RACFIND=? LOG=NOSTAT CODES=00/00

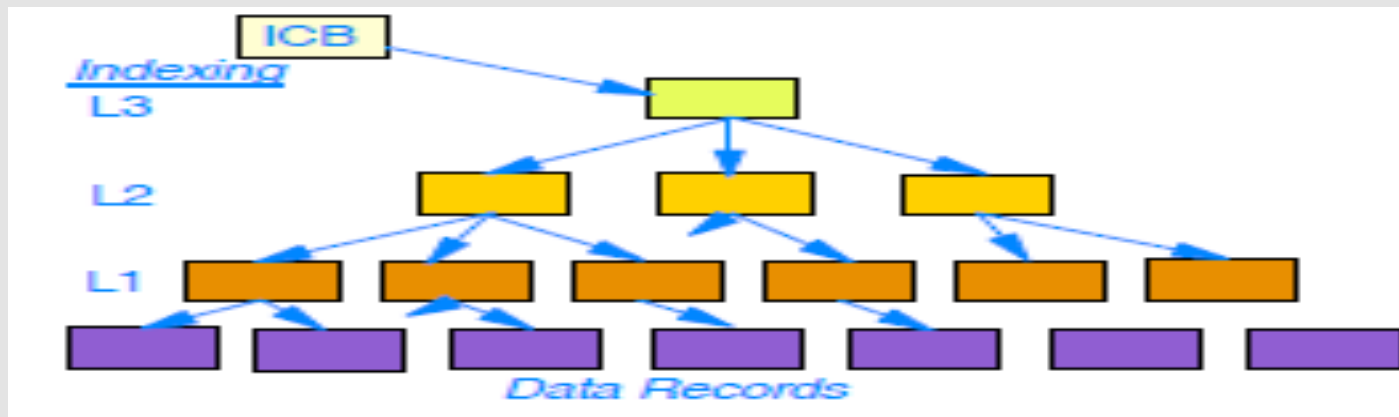
RACF in action

- Development system: 120 RACF calls/second
- Average time to complete: 1.654 milliseconds

RACF Processing: Logical dataset

- Multilevel index set to locate profiles:
 - ICB (Inventory Control Block)
 - L3
 - L2
 - L1 (sequence set)
 - Data (entity records) contain RACF profiles
- 4K Blocks addressed by their RBA (Relative byte address).

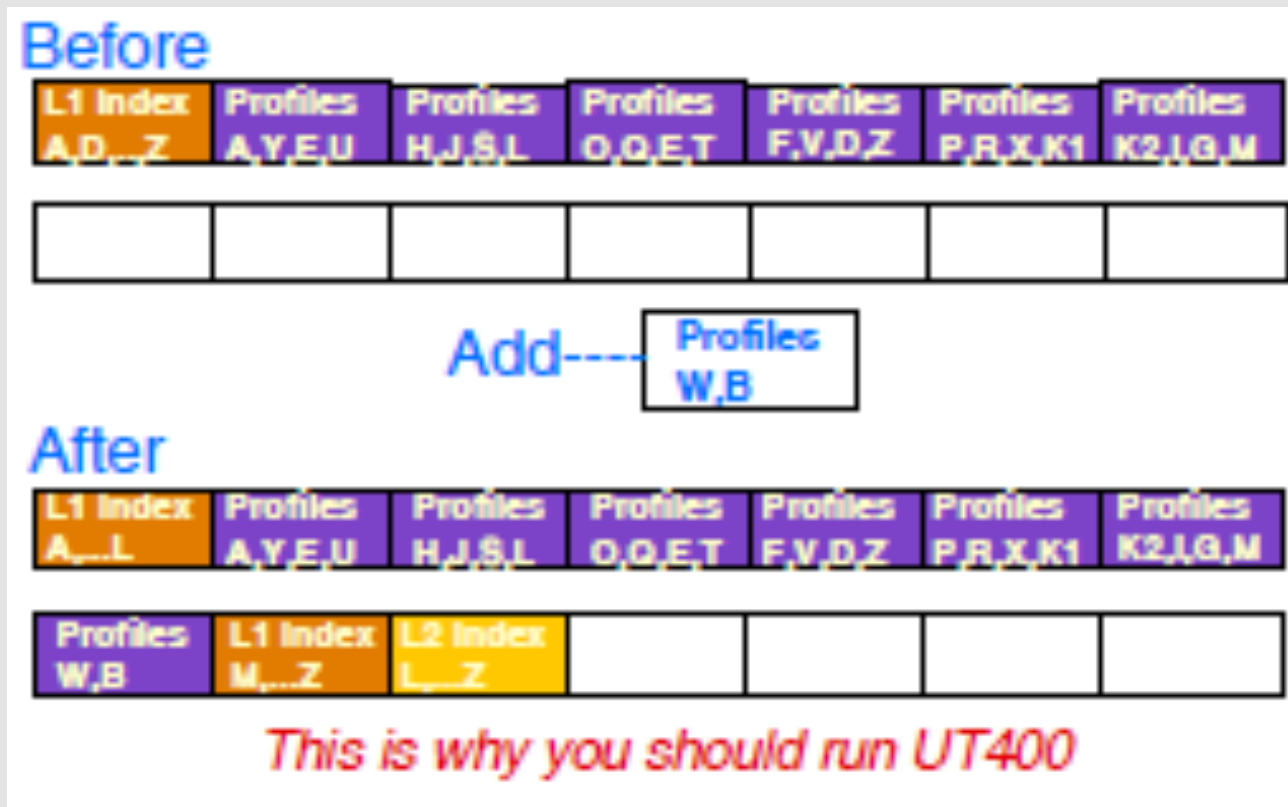
Index structure



RACF Processing: Types of blocks

- Templates & segment table blocks contain DSECTs for the layout of data record
- ICB contains:
 - the first index pointer
 - the RACF options
- The ICB is READ regularly by all systems in a sysplex when not in datasharing mode, otherwise RACF uses XCF to communicate the changes
- BAM Blocks contain a description of the space in a RACF database

Defragmentation



Defragmentation

- Over time Indices and data blocks gets separated
- Entity records can be split over several datablocks
- The solution: IRRUT400
 - Makes a logical copy of your RACF database
 - Can be used to split (merge) a RACF database
 - Do not confuse with IRRUT200
 - Should be executed regularly

Statistics

- IRRUT200 is a standard RACF utility that produces statistics
- Run it regularly and look for anomalies

Tuning Options: I/O Considerations

- Shared DASD
 - Easy to maintain a consistent view
 - Cost is RESERVE or global ENQ
 - The ICB is read by every system regularly

Tuning Options: I/O Considerations

- Shared DASD alternative (seldom used)
 - Create copies of the RACF database(s) and assign one to every LPAR in the Sysplex
 - Maintain the different copies synchronized with the RACF Remote Sharing Facility
 - Disadvantages:
 - Some RACF commands are not propagated by RRSF
 - By using the RACF subsystem they could be propagated by executing them as operator commands
 - If catalogs are shared in a Sysplex, every LPAR must run with a different copy of ICHRDSNT (The RACF database name table) & eventually the RACF Range table.

Tuning Options: I/O Considerations

■ Statistics

- There is an option in the RACF database name table that avoids that statistics are written to the backup database(s)
- If the SETROPTS INITSTAT is on, a limited subset of statistics will be written to the backup database.
- Recommended is not to write statistics to the backup DB

Tuning Options: Audit Considerations

■ SMF

- RACF writes SMF records when requested and so can generate a lot of I/O
- Most of these are seldom really used
- Check on a regular base what is recorded and whether you need it or not

Tuning Options: Audit Considerations

- Commands that affect auditing:
 - APPLAUDIT or NOAPPLAUDIT
 - AUDIT or NOAUDIT
 - CMDVIOL or NOCMDVIOL
 - LOGOPTIONS
 - OPERAUDIT or NOOPERAUDIT
 - SAUDIT or NOSAUDIT
 - SECLABELAUDIT or NOSECLABELAUDIT
 - SECLEVELAUDIT or NOSECLEVELAUDIT

Tuning Options: RACF Set-up

■ Resident data blocks

- Again in the database name table there is a byte that tells RACF how many 4K blocks it must cache is ECSA.
- If you do not use a database name table this can be set from the loadmodule ICHSECOP
- If you split your RACF DB, this number can be specified for every DB

Tuning Options: RACF Set-up

- Global access table
 - Profiles can be defined in the class GLOBAL with a UACC
 - During NIP, RACF copies this into storage
 - When asked to do authority checking, RACF will always first look in this table and when it can grant access it will do it without I/O
 - We measured RACF processing time of 0.04 milliseconds for a DATASET profile in the table while the average is 1.6 ms

Tuning Options: RACF Set-up

- Global access table caveats
 - If the access in the RACF DB is lower than the one in the table, access will be granted, an example:
 - Table contains DATASET SYS1.* / READ
 - DB contains DATASET SYS1.PARMLIB ID(jan) access NONE
 - Jan has READ access on SYS1.PARMLIB
 - No logging and no statistics

Tuning Options: RACF Set-up

■ Global access table candidates

• DATASET

- SYS1.*/READ
- SYS1.BROADCAST/UPDATE
- All (ISPF & SDSF, ... DATASETS).*/READ
- Master catalog READ
- User catalogs UPDATE

• JESJOBS

- SUBMIT.*.&RACUID*.&RACUID/READ

• JESSPOOL

- *.&RACUID.* /ALTER (G)
- *.*.&RACUID*.* /ALTER

• SERVAUTH

- EZB.STACKACCESS.SYT%.NCTCP*/UPDAT

Tuning Options: RACF Set-up

- Global access table recommendations
 - Scan for uACC different from NONE profiles and put them in GLOBAL
 - Calculate the size in ECSA:
 - $27\ 640 + 2\ \text{Pts} (18 + \text{number_of_entries Pts} (6 + (1.5\ \text{Pts max_profile_name_size})))$
 - Check not to undermine a lower level specification

USS

- stage 3 of application identity mapping (AIM)
 - Eliminates:
 - UNIXMAP class
 - VLF IRRUMAP, IRRGMAP a IRRSMAP
 - Works with a 'new' alias index in the RACF DB
 - Gives performance benefits for USS processing

Generic profiles checking (1)

- When a dataset is opened for the first time in an A.S., RACF will cache an index with a list of all the profiles behind the HLQ in LSQA
- With a maximum of 4 entries (HLQ's pointing to the profiles), the list is maintained on a FIFO base
- The CPU needed to charge this list of for every HLQ depends on the number of profiles behind the HLQ
- If there are more than 500 profiles behind a HLQ the time to load them becomes 'uncontrollable'

Generic profiles checking (2)

■ Example:

- Job starts with 4 STEPLIB datasets:
 - SYS1.something
 - SYS2.something
 - IMS.something
 - CICS.something
 - At this point, the A.S. contains the profiles behind SYS1, SYS2, IMS & CICS
- Jobs opens 4 files:
 - APPL1.something, ... APPL4.something
 - At this point the A.S. contains the profiles behind APPL1, ... APPL4
- Job step 2 has the same steplib
 - APPL1, ... APPL4 are removed and SYS1, SYS2, IMS & CICS are rebuild

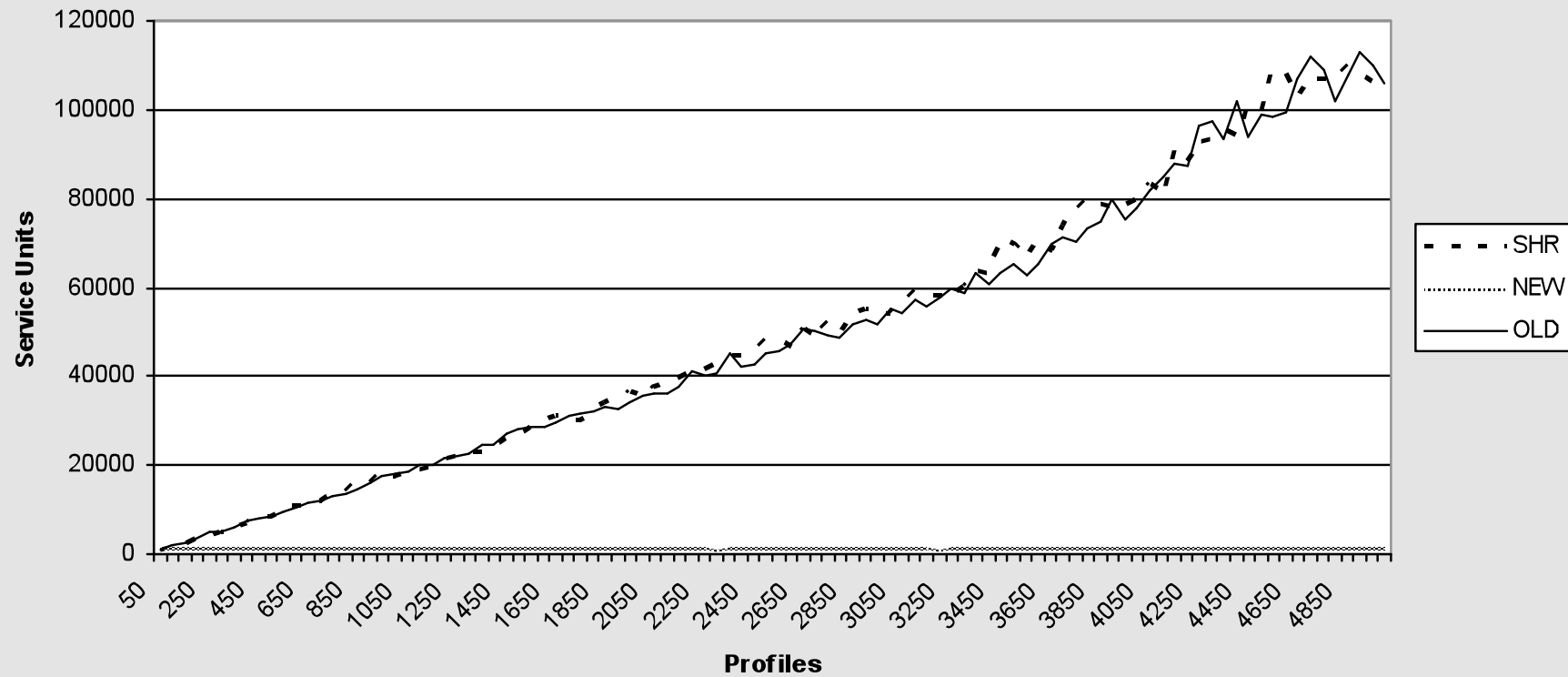
HLQ's and Service units (1)

- Open a file

Profiles	Service Units
143	3658
2486	47968
4617	78950
5830	80013

HLQ's and Service units (2)

RACF & Disposition



HLQ's and Service units (3)

- Recommendations:
 - Avoid JOB- & STEPLIB
 - Redesign naming conventions (eventually with naming convention exit) if possible
 - Avoid SETROPTS GENERIC REFRESH because this causes the LSQA HLQ caching to be invalidated

Tuning Options: RACF Set-up

- An ACEE is the representation of a user in an address space.
- This information can be kept in VLF:
 - VLF is searched before the RACF DB's
 - Update the COFVLFxx member of SYS1.PARMLIB as follows:


```
CLASS NAME(IRRACEE)    /* RACF ACEE Data in Memory */
EMAJ (ACEE)            /* Major name = ACEE
```
 - check for installation use of the ACEEIEP field
 - Logon, batch submission, APPC & CICS Reconnect will improve

Tuning Options: Split the RACF DB

- IRRUT400 can split a DB the advantage is not only that there is spread of the I/O but also that the number of resident index blocks can be increased. The latter is a performance booster.
- Where to split?
 - With a database manager trace and some Rexx to read the RACF database and to analyse the GTF trace, an optimal split can be calculated.

Tuning Options: RACLIST

- For a RACLISTED class the profiles are kept in a DATASPACE, accessible for all A.S. No I/O is ever done apart during NIP & when the dataspace is refreshed.
- Not every class can be RACLISTED (DATASET for instance not), this is specified in the Class Descriptor Table (CDT)

Tuning Options: RACGLIST

- 'Pages' a dataspace in that contains RACLISTED profiles from the RACF database:
 - During RACF NIP
 - After a refresh (first image that does the refresh pages out)
- Pages are kept in the RACGLIST class profiles
- Advantage is that systems in a Sysplex only have to page in the profiles after the first one constructed them
- XCF is required
- Caveat: The RACF database could become too small

RACF organisation

- Group tree in storage:
 - Uses VLF class IRRGTS
 - Saves information for users with:
 - Group-SPECIAL
 - Group-OPERATIONS
 - Group-Auditor
 - Do not use with split RACF DB's
 - In general not recommended

RACF Administration

- RACF commands can cause a lot of profiles to be read:
 - SEARCH
 - LISTDSD with the ID or PREFIX operands
 - LISTGRP *
 - LISTUSER *
 - RLIST classname *
- This can in some cases lead to contention on the RACF DB

RACF Administration

- IRRDBU00 is a RACF utility that unloads the DB to a sequential file.
- Most sites run this every night.
- Use the unloaded data to process large queries

RACF Enqueue

- RACF uses RESERVE/RELEASE for every profile that it reads when:
 - not enabled for sysplex communication
 - enabled for sysplex communication but the system is running in non-data sharing mode
 - It can not find the data in the resident index blocks
- To convert the RESERVEs, place a generic entry for SYSZRACF in the RESERVE conversion resource name list (RNL) in your GRSRNLxx (PARMLIB)

RACF Enqueue

- If a user is swapped out while holding an exclusive enqueue on the RACF major name, other RACF activity can be blocked
- Let the user finish the task by increasing the enqueue residency (ERV parameter in IEAOPT):
 - Default = 500
 - Recommended 4000 – 50000

The coupling facility & XCF

- XCF propagates commands like RVARY & SETROPTS REFRESH if set by a bit in ICHRDSNT
- Datasharing mode uses a structure in the coupling facility to cache RACF profiles inside a sysplex
 - A profile found in the CF does not require:
 - I/O
 - ENQUEUE processing
 - Rule of thumb: reserve space in the coupling facility for all profiles for all classes that are not RACLISTED (the minima are probably too small)

RACF Administration

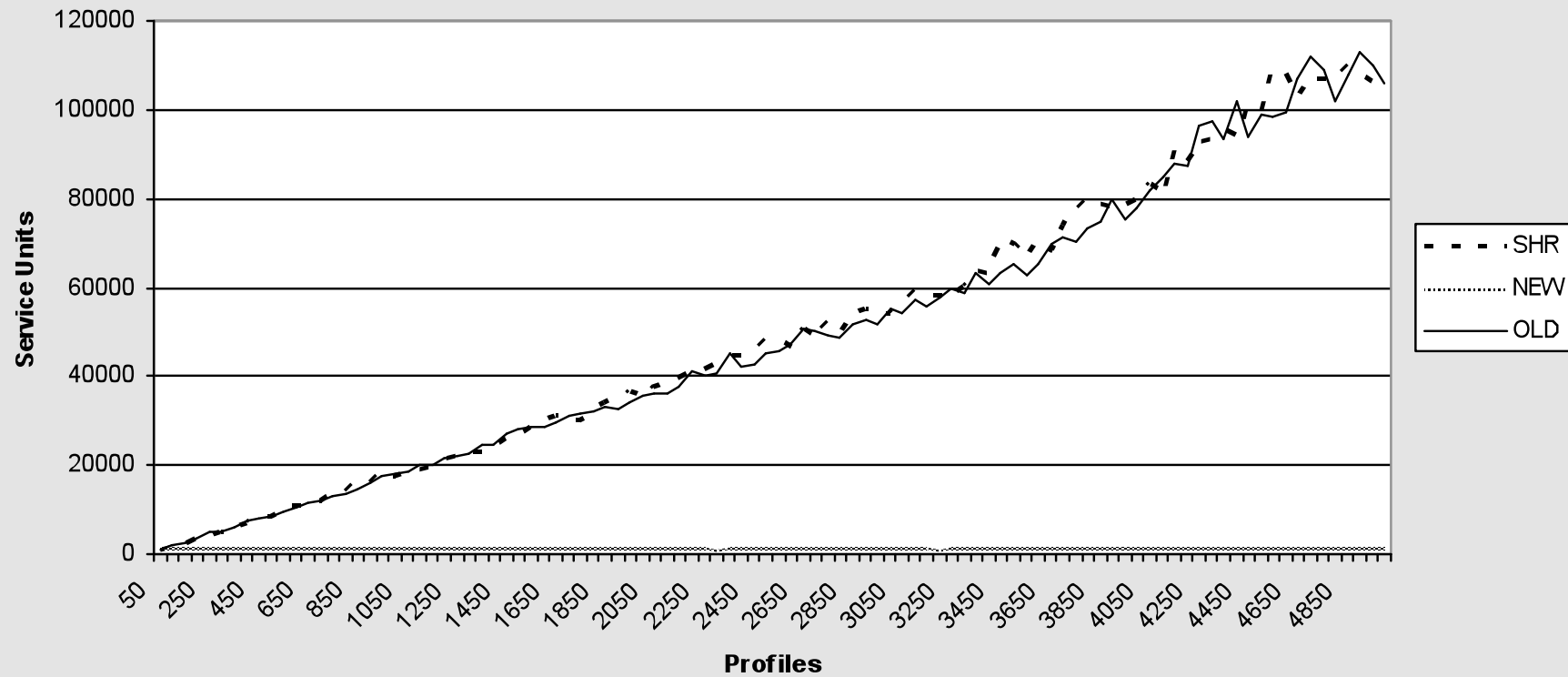
- End user response time goes down with:
 - Erase on scratch, an option whereby a dataset is overwritten with '01'x when deleted
 - Very large:
 - Access lists
 - Groups
 - Number of connections
 - Long group trees

RACF Administration

- Avoid redundant entries in the RACF DB
 - Run (eventually a trial of) CONSUL/RACF or VANGUARD
 - Check things like:
 - Is a TSO logon procedure in RACF still existing in the proclib?
 - Do we need account numbers?
 - Run the remove ID utility
 - Is the user/group access equal to the UACC

HLQ's and Service units

RACF & Disposition



Recommendations summary (1)

- Run IRRUT400 on a regular base
- Run IRRUT200 without the copy function from time to time
- Do not write statistics to the RACF backup database(s)
- Put in GLOBAL what you can
- Split the RACF DB's
- RACLIST everything you can
- Put the ACEE's in VLF

Recommendations summary (2)

- Convert the RESERVES to GLOBAL ENQUEUEES
- If possible work on a RACF unloaded DB for queries
- RACGLIST what you can in a Sysplex but beware of the space requirements
- Without split RACF DB's and with many special attributes on the group level you might consider GROUP TREE IN STORAGE
- Sizing the RACF structure in the coupling facility can avoid I/O contention

Recommendations summary (3)

- Avoid erase-on-scratch
- Avoid large groups, large number of connections for a user, large access lists and long group trees
- Check that you are running stage 3 of application identity mapping
- Avoid redundant entries in the RACF DB
- Review the profiles/HLQ situation
- Check the value of ERV in IEAOPT