IBM Software Group

ALCS Allocatable Pool

alcs@uk.ibm.com

Agenda

- Review of original Long Term Pool
- Migration of Long Term Pool to Allocatable Pool (Type 2)

- Recoup messages

Once upon a time a long time ago …

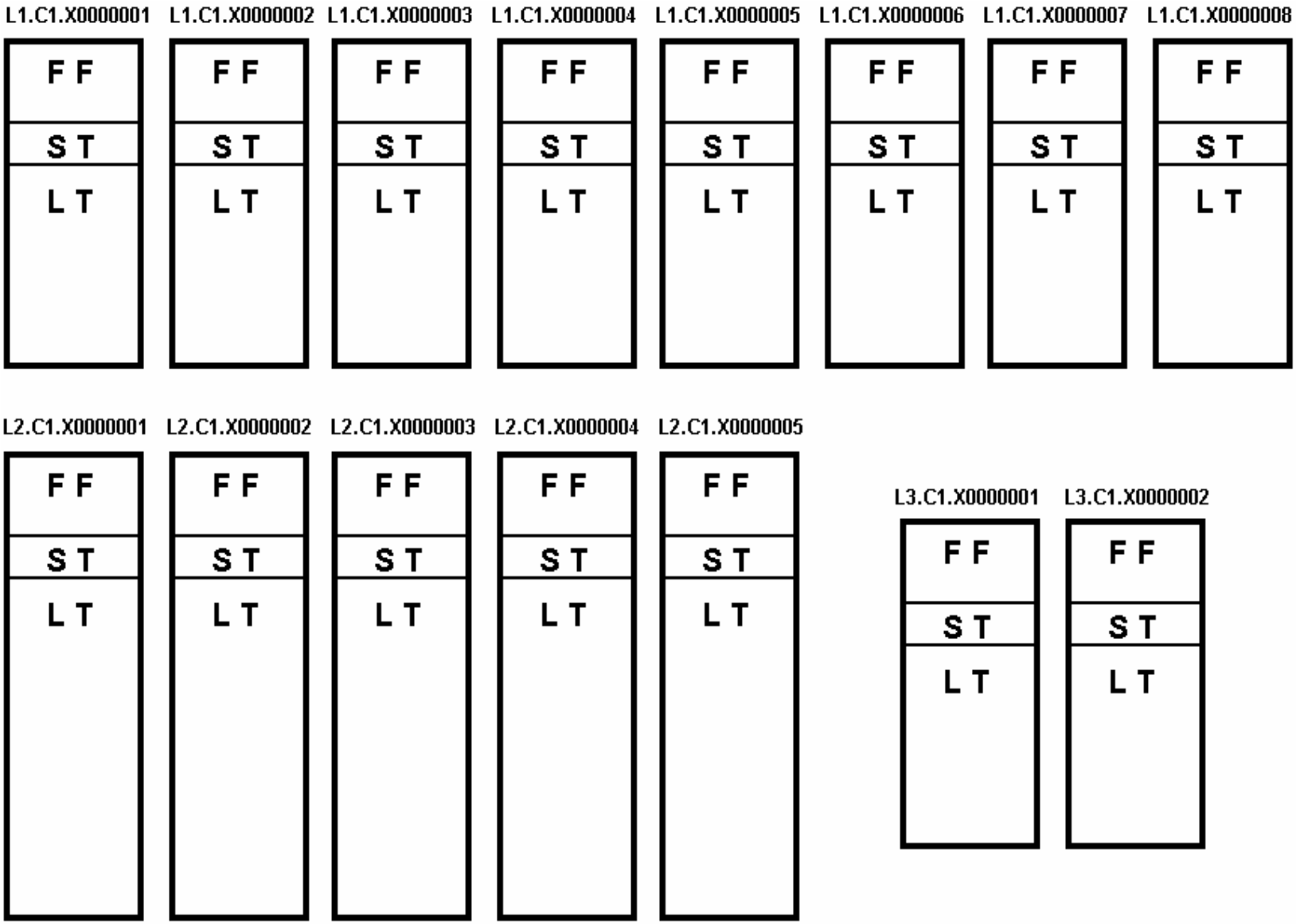
- ALCS Version 1 and ALCS Version 2.1.1
- Describe
  - Database
  - Database Generation
  - ZDATA DUMP / LOAD
  - Database expansion

Real-time Database

Collection of datasets on disk

- Each is a VSAM single extent Entry Sequenced Data Set (ESDS)
  - Each record size (L1, L2, L3, …) has it's own data sets
    - Containing fixed file and pool file records
    - Records are 'striped' across the datasets to balance I/O load
  - Data set block size is a little bigger than the ALCS record size
    - One record per block

	Application	System
L1	381	512
L2	1055	1536
L3	4000	4096
L4	4095	4608


L1.C1.X0000001
L1.C1.X0000002
L1.C1.X0000003
L1.C1.X0000004
L1.C1.X0000005
L1.C1.X0000006
L1.C1.X0000007
L1.C1.X0000008
F F
S T
L T
L2.C1.X0000001
L2.C1.X0000002
L2.C1.X0000003
L2.C1.X0000004
L2.C1.X0000005
L3.C1.X0000001
L3.C1.X0000002

Record Identification

- Each database record has a 4 byte file address made up of
  - BAND (represents the file type)
    - #WAARI, #PNDRI, L1ST, L1LT
  - ORDINAL
    - Record number with type. #WAARI(317), L1LT(625778)

Database Generation Statements

DBGEN …

GFGEN …

…

USRDTA …,TYPE=#ABCDE,BAND=4001,SIZE=L1,NUMBER=1000,…

USRDTA …,TYPE=#XYZ12,BAND=4002,SIZE=L1,NUMBER=500,…

USRDTA …,TYPE=#XYZ13,BAND=4003,SIZE=L1,NUMBER=70,…

USRDTA …,TYPE=#XYZIX,BAND=4004,SIZE=L1,NUMBER=2,…

USRDTA …,TYPE=#AAREC,BAND=4004,SIZE=L1,NUMBER=30,…

….

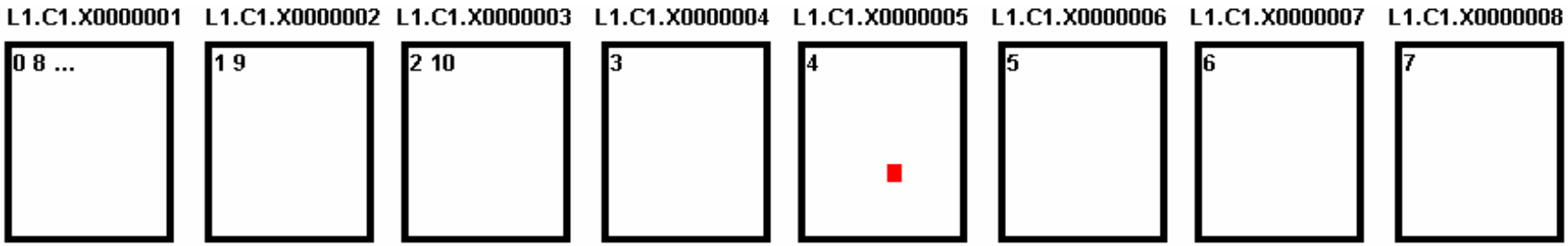
…

#AAREC(0)

The generation stage 2 shows the information used for the database control blocks

```
#AAREC          DF0DT    ACTION=ENTRY,SIZE=L1,NUMBER=(30,30),
                         …,TYPE=FIXED, BASE=1572
```

The entry indicates that ordinal 1572 in all the L1 records is the first #AAREC. Suppose there are 8 L1 datasets. Then ALCS can used the above information to compute which dataset and relative record within the dataset holds the record for the file address



1572/8 = 196 remainder 4 so the 197th Block in the 5th dataset

Record Identification

- Access to each record in the database is by VSAM call passing :
  - Data Set Name
  - Sequence of the record within the Data Set (Relative Record Number)
- As we saw ALCS computes the above from the file address using an simple ALGORITHM so we now call this :

ALGORITHM BASED ADDRESSING

ZDATA DUMP / LOAD

- ZDATA DUMP
  - Dump records from the real-time database to sequential file
  - Writes each record prefixed by it's file address
- ZDATA LOAD
  - Load records from sequential file to the real-time database
    - System must be in IDLE state
- ZRSTR
  - Restore the DASD record updates from the ALCS update log file
    - System must be in IDLE state

Data base expansion

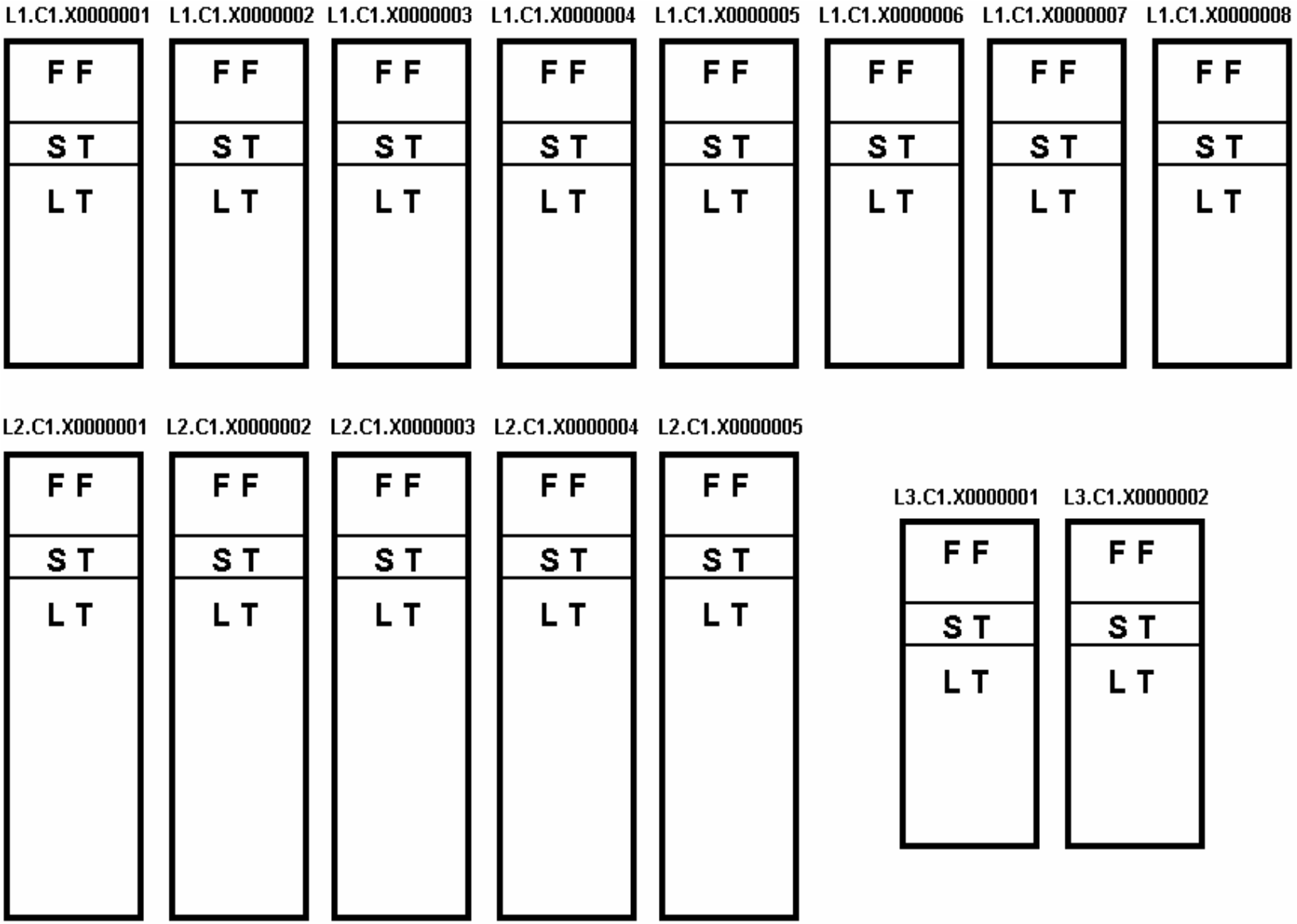
- ZDATA DUMP
  - Dump records from the real-time database to sequential file
- Use new (enlarged LT pool) database generation
- ZDATA LOAD
  - Load records from sequential file to the real-time database
    - System must be in IDLE state
- ZRSTR
  - Restore the DASD record updates from the ALCS update log file
    - System must be in IDLE state

Data base expansion

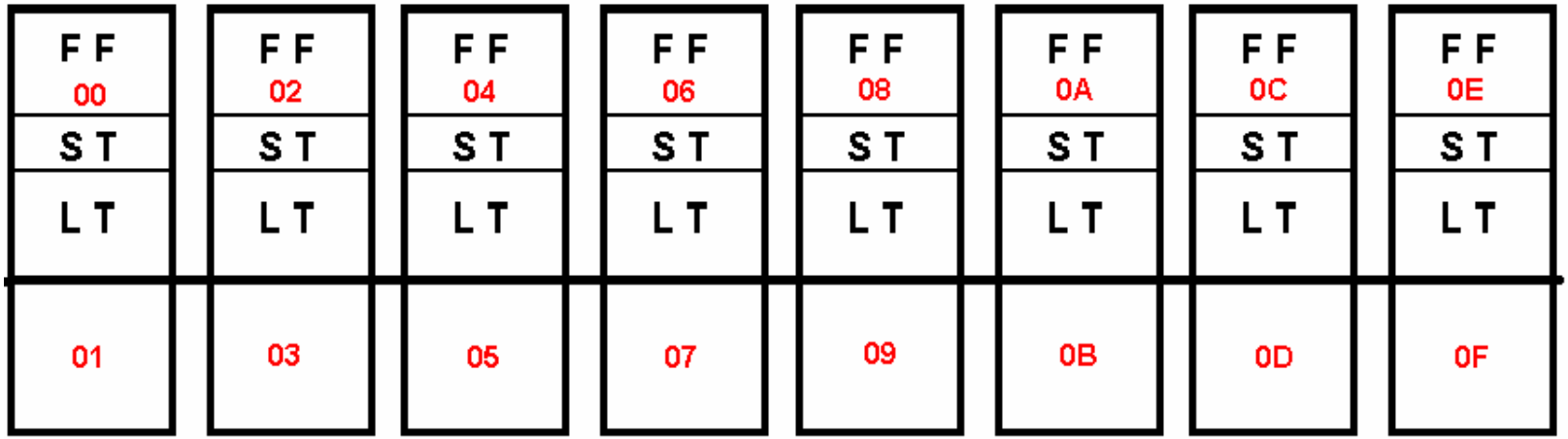
- Involves a long outage
  - Even when building new database in parallel
    - Need to checkout new system
- Months of planning and testing
  - Impacts other projects

ALCS 2.1.3

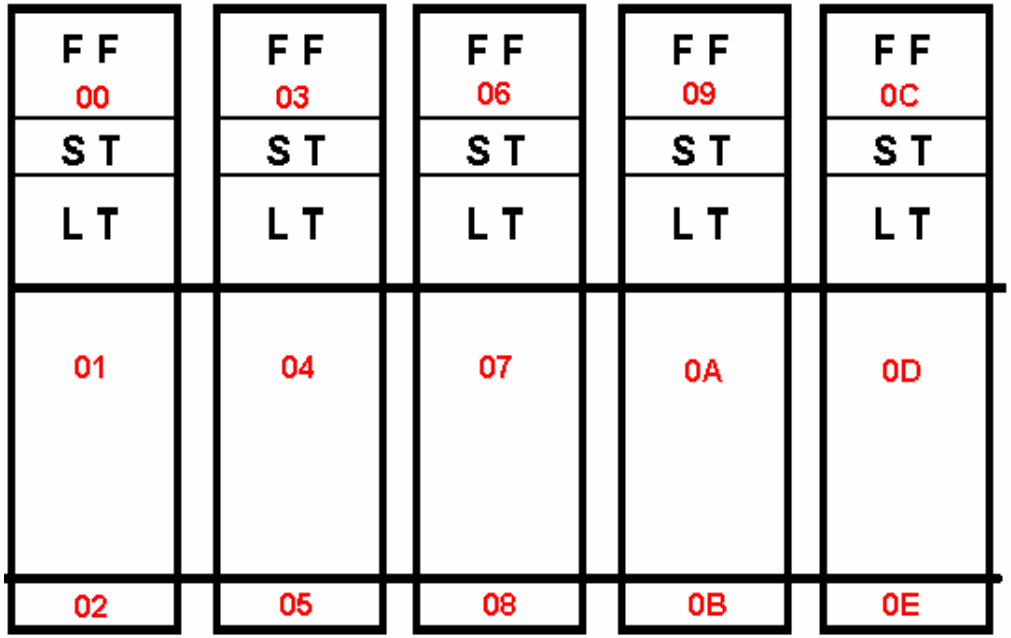
- DBGEN - new BAND= parameter
- Each record in database has two file addresses, the
  - Previous Algorithmic File Address, and the
  - New Allocatable File Address
    - Uses new BAND from the DBGEN to indicate size
    - With segment and ordinal based upon physical position in the database


L1.C1.X0000001
L1.C1.X0000002
L1.C1.X0000003
L1.C1.X0000004
L1.C1.X0000005
L1.C1.X0000006
L1.C1.X0000007
L1.C1.X0000008
F F
S T
L T
L2.C1.X0000001
L2.C1.X0000002
L2.C1.X0000003
L2.C1.X0000004
L2.C1.X0000005
L3.C1.X0000001
L3.C1.X0000002

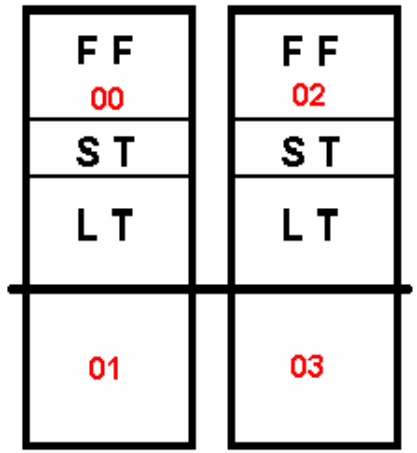
POOL SIZE L1 EACH DATASET HAS 100000 RECORDS SO 2 SEGMENTS of 64K RECS EACH



POOL SIZE L2 EACH DATASET HAS 150000 RECORDS SO 3 SEGMENTS OF 64K RECORDS EACH



POOL SIZE L3. DATASETS OF 100000 RECORDS. SO 2 SEGMENTS TO EACH



Allocatable File Address

- For example suppose the new BAND for size L1 pool is x'D1' and you have eight L1 datasets each with 100,000 records. Then you will have addressability for 16 segments (00 to 0F in hex) two in each dataset (as 100000 is larger than 64K but less than 2 x 64K).
- The first record in each dataset would be 000000D1, 020000D1, 040000D1, ... , 0E0000D1
- The last record in each dataset would be 01869FD1, 03869FD1, 05869FD1, ... , 0F869FD1
- If in the DBGEN the fixed file records come first followed by the short term pool followed by the long term pool then 01869FD1 will be a long term pool record and the 020000D1 will be a fixed record.

Migration of long term pool to Type 2

The migration of Long Term Pool to Type 2 :

- 1. Create a DASD generation that includes a DBHIST macro with:

  DISPENSE_TYPE2_LONG_TERM

- 2. Load, confirm, and commit the new DASD generation (ZDASD command)

- 3. Run Recoup

Creates Type 2 directories and ALCS dispenses allocatable pool file addresses.

There are 2 things to ensure before migration

1. Your applications are capable of handling both types of file addresses.
   - Beware programs that interpret / convert file addresses.
     - IPARS PRC1 converts file addresses to 6 character PNR locators. So need to ensure no overlap between the PNR locators generated from the old style and new style file addresses.

2. Alter any processes that scan long term pool sequentially.
   - Some users have batch type jobs that go through all the pool records by ordinal.
     - With the old addresses this I/O would be evenly spread across all the datasets
     - With the new addresses it will result in sequential retrieval from each dataset in turn
     - Use program CAP1 to retrieve the records you wish.

Both the above items require testing so migrate a copy test system and let the application programmers test to see if the dispensed addresses can be handled correctly.

File Address Lookup Tables

- Non-algorithmic addresses are looked up in tables
  - Reference type/ordinal to allocatable pool address for new ST pool and FF
- The tables have a three level tree structure for fast lookup
  - Top level index is stored in System Fixed File (L3)
  - Lower levels in System Used allocatable pool (L3)
  - Held in-core in MVS Dataspaces. Not paged
- The tables exist logically to hold the address of every record but in reality are only populated if the file addresses are not algorithmic
  - You may choose to build type 2 directories for a whole file type
  - You may choose to build type 2 directories for all file types

Adding FF & ST

- Generate a new DASD configuration with

  USRDTA ACTION=ADD,TYPE=…..,NUMBER=100

- For Fixed File, the effect is immediate on first use:
  - When the fixed record is requested, a LT pool record is dispensed
  - The File Address Lookup Table is updated to reflect it
- For Short Term Pool a RECOUP run is required:
  - To allocate and file down the new ST pool records
  - The File Address Lookup Table is updated to reflect them

Deleting FF & ST Pool records

Deleting records is a two stage operation

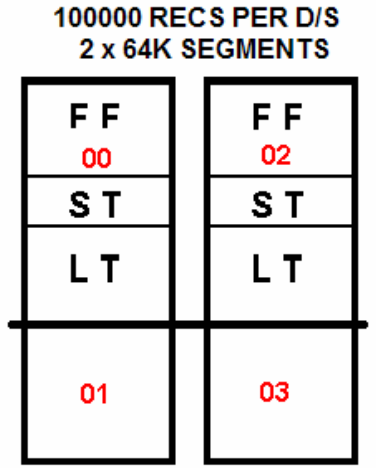
1) Update Data Base generation with:
   - DBHIST UPDATE
   - USRDTA ....,ACTION=DELETE
   - ZDASD LOAD ⇨ ZDASD CONfirm ⇨ ZDASD COMmit

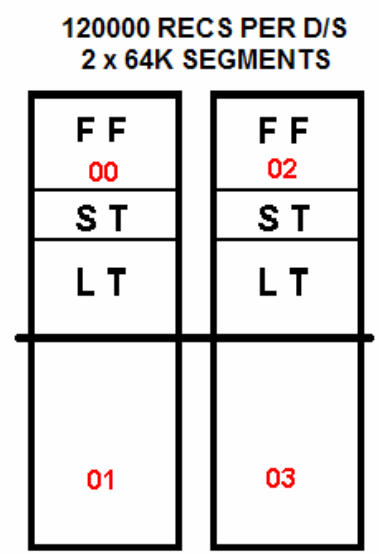
2) Update Data Base generation with:
   - DBHIST UPDATE
   - USRDTA ....,ACTION=PURGE
   - ZDASD LOAD ⇨ ZDASD CONfirm ⇨ ZDASD BACKOUT or COMMIT

Expanding pool - 1

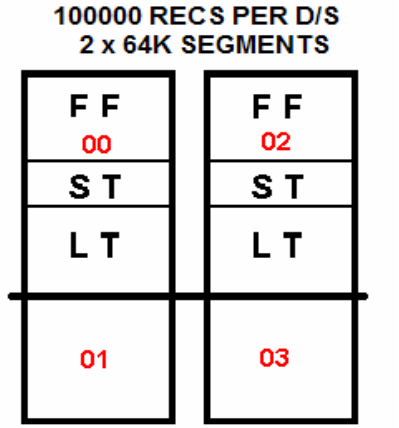
Within the present logical addressability

- Example. Datasets of 100,000 expanded to 120,000 records. This is done by increasing the size of the datasets.
- To do this:
  - Vary off the copy 2 dataset.
  - Delete and redefine a larger copy 2 dataset.
  - Vary on the copy 2 dataset, automatic copy will ensue
  - Vary off the copy 1 dataset.
  - Delete and redefine a copy 1 similar to copy 2 dataset.
  - Vary on the new copy 1 dataset
  - At the next Recoup, the added records will become available to allocatable pool.





Expanding pool - 2



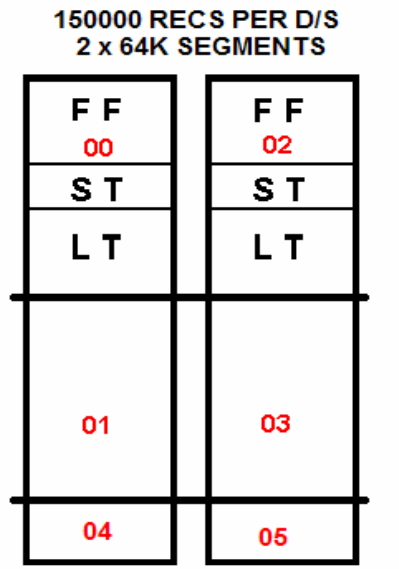
Extending the logical addressability

- Example. Datasets of 100,000 expanded to 150,000 records. Each dataset requires an extra 64K segment
- To do this:
  - Generate a new DASD configuration with

    DBSPACE SIZE=Lx,SEGMENTS=1

    Segments is the number of additional segments to add to each data set. Each data set is allocated the same amount of additional addressability
  - Then increase the physical size as shown previously
  - The actual size, number of records, in the data sets can be less than the amount of logically-addressable records

    In our example 150000 < 196608



Expanding pool - 3

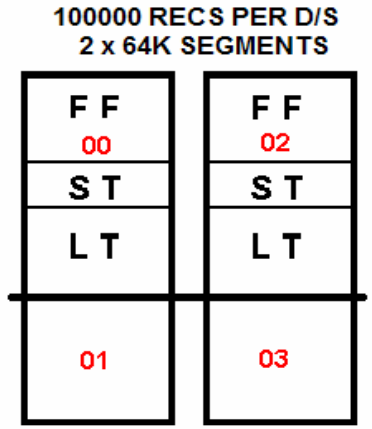
Extending the logical addressability

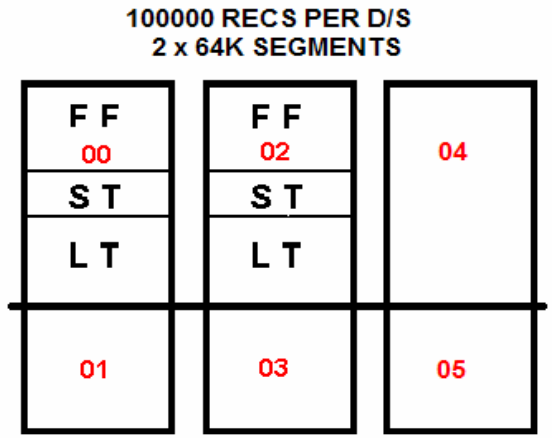
- Example. Adding a dataset of a 100,000 records.
- To do this:
  - Generate a new DASD configuration with

    DBSPACE SIZE=Lx,VOLUMES=1

    Volumes is the number of additional data sets to make addressable for the record size. Each data set is allocated the same amount of addressability as the existing data sets for this record size.
  - Then increase the physical size as shown previously
  - The actual size, number of records, in the data sets can be less than the amount of logically-addressable records

    In our example 100000 < 131072

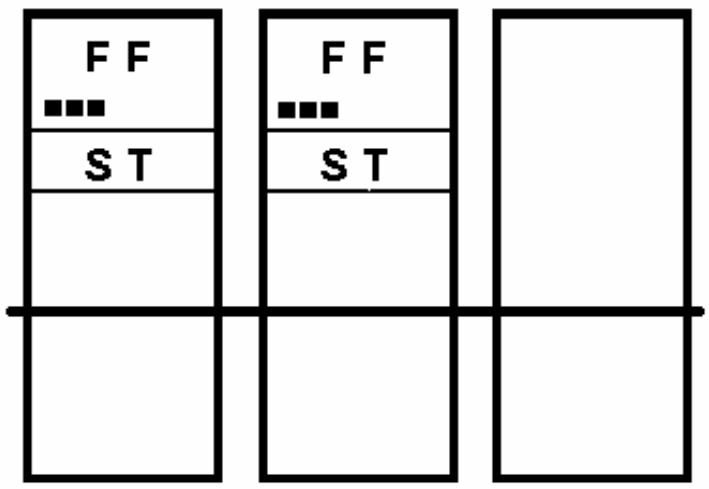


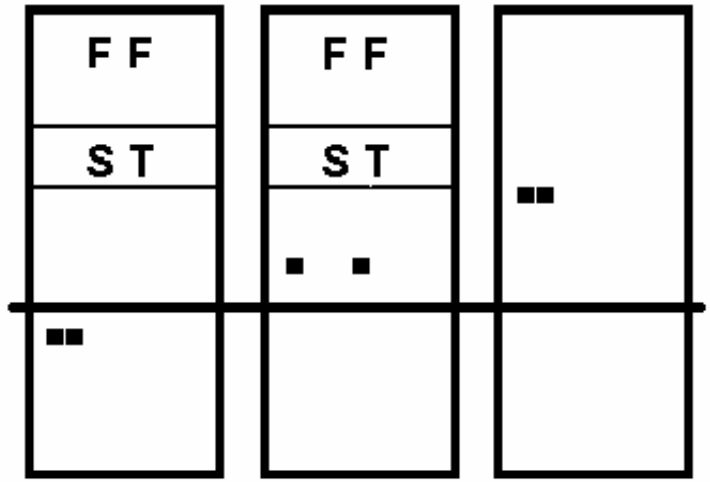


Converting a file type to table based addressing

- Activate table-based addressing through the DASD configuration
  - For the specific record types that will be restriped, use USRDTA parameter
    - USRDTA TYPE=type,ACTION=BUILD
  - For all fixed file and short-term pool records use DBHIST parameter
    - DBHIST BUILD_DIRECTORIES,TEXT='.....'
  - Both take effect after ZDASD COMMIT of the Load
- Table size: 1 million allocatable records requires just over 4MB of dataspace. Include this storage in the working set size for ALCS.

Restripe





Restripe

- Activate table-based addressing for the record types to be redistributed:
- For example, for #WAARI fixed records and L2ST pool:
  - Code USRDTA macros in the DASD generation
    - USRDTA TYPE=#WAARI,ACTION=BUILD
    - USRDTA TYPE=L2ST,ACTION=BUILD
  - Online load and commit the DASD generation

- Activate restripe for the fixed records
  - ZDASD STRIPE,#WAARI
- Activate restripe for the L2 short-term pool
  - ZDASD STRIPE,L2ST

Restripe Processing

ALCS performs the following for each record which is restriped

- Obtains an available long-term pool record (allocatable pool address)
- Moves record to its new location on the data base:
- reads record at its current file address
- writes record to the allocatable pool address

- Accesses the file address lookup tables:
  - finds slot for the record being redistributed
  - inserts allocatable pool address into slot

- Updates status in the Restripe Keypoint record

RECOUP - Allocatable pool directories

- ALCS uses one dataspace for each allocatable pool
- During RECOUP ALCS uses two dataspaces for each allocatable pool

- Each dataspace requires approximately one bit for each record in the corresponding allocatable pool

RECOUP – end messages

DXC8370I CMD W 16.05.38 RECP Chain chase complete

CHAIN CHASE ELAPSED TIME 000.04.43

DXC8409I CMD W 16.05.38 RECP

Recoup directory build -- Fixed file / ST pool scan in progress

DXC8408I CMD W 16.06.11 RECP

Recoup directory build -- Recoup general file scan in progress

DXC8371I CMD W 16.06.12 RECP Directory build complete

RECOUP – end messages

DXC8408I CMD M 18.47.08 RECP

Recoup directory build -- Recoup general file scan in progress

DXC8399W CMD M 18.47.08 RECP Directory build ** BAD ** -- GF-000 i/o error, full or off-line

DXC8371I CMD M 18.47.08 RECP Directory build complete

Ensure PRIMECNT and COUNT are big enough

When they are just exceeded a warning is issued

DXC8378W CMD W 18.14.09 RECP

Too many reads this prime group -- Prime group continues

… but if this is ignored and the number of records grow over time …

Records will be missed

DXC8378W CMD W 18.49.01 RECP

Too many reads this prime group -- Prime group continues

DXC8378W CMD W 18.49.06 RECP

Too many reads this prime group -- Prime group continues

DXC8393E CMD W 18.49.13 RECP

Too many reads this group -- Group terminated