z/TPF Detailed Summary

System Characteristics

Mark Lehrer
z/TPF Development

System Characteristics Agenda:

Entry Control Blocks

Message Processing

Message Flow

Memory Configuration

Main Storage Management

System Characteristics Agenda:

Entry Control Blocks

Message Processing

Message Flow

Memory Configuration

Main Storage Management

Entry Control Blocks
Note on “CPU” terminology

Since TPF was initially developed for single-CPU systems, some terminology has diverged from common IBM architecture terminology.

In z/Architecture terms, a “CPU” contains the sequencing and processing facilities for instruction execution, interruption action, and timing functions; each CPU contains its own general registers, control registers, and PSW; and multiple CPUs in a system can run concurrently.

The TPF term for a z/Architecture CPU is an instruction stream, or I-stream.

- Note that some of the more recent TPF documentation uses the term CPU in the z/Architecture sense, especially when related to hardware features.

In z/Architecture terms, a “system” includes main storage, one or more CPUs, the channel subsystem, and I/O devices.

TPF often refers to a z/Architecture system using the term CPU; each CPU is given a single alphanumeric character as an identifier called a CPU ID.

Entry Control Blocks
Entry Control Blocks (ECB)

Entry
- An entry is associated with an input message from an end user.
- For entries to be processed concurrently, an area of main storage is assigned to each entry for use as both an application work area and a system work area.
- This area of main storage is called an **entry control block**, or commonly called an **ECB.**

The **ECB** is the cornerstone of the application program **re-entrant** structure.

Use of the ECB allows one program to service multiple entries because **each ECB is private to an entry.**

Note that while an “entry” is typically considered to be synonymous with an input message, it is more generically associated with a “unit of work”
- An end-user input message
- A command from an operator
- A routine initiated by a time-initiated utility
- A task “spun off” from another task to be executed asynchronously

Entry Control Blocks
Entry Control Blocks (ECB) - continued

Linked to a task (typically an input message but can be almost any work unit)

Maintains a record and reference area of resources used.

Serves as the interface between the TPF system and the application, and between other ECB-controlled programs within the application package.

Provides pre-allocated storage for system use and application use.

Each message creates a new ECB.

One program can be used by many ECBs.
- Application programs are required to be re-entrant (cannot be self-modifying).

The ECB model has characteristics of both a process and a thread.
- The footprint of an ECB is lightweight like that of a thread
- The ECB still exhibits many characteristics of a process
  - Each ECB has its own address space, stack area, heap storage, etc

Entry Control Blocks
Entry Control Blocks (ECB) - continued

Within the ECB, there are work areas that are an integral part of a non-reentrant program.
- **EBW000-EBW103** is the primary work area of the ECB.
  - Can be used to pass data from one program to another or as a scratchpad area for calculations.
- The following 8 bytes are typically inter-program bit switches.
  - The inter-program switches are initialized to zero when the ECB is created.
- The secondary work area (EBX000-EBX103) uses the same conventions.

The ECB is allocated from main storage blocks during IPL.

Portions of the ECB are defined as part of application-programming interface (API)

Entry Control Blocks
ECB Format

ECBs are allocated at IPL time. The number is specified by a user-defined SIP variable.

The ECB is a 12K-byte block and is composed of 3 4K-byte pages.

ECB page 1

- Used primarily by applications.
- Contains fixed application work areas.
  - Used by the application program. The work areas are of fixed sizes, and their use is user-defined.
- Contains fixed system work areas, which are comprised of:
  - A resource-interface area used by both the application program and the control program. This area contains the addresses of file records and working storage blocks that the application program has requested from the z/TPF system.
  - An entry-management area used by the control program. This area contains status information and other data required for managing an entry.
  - A user-definable area which can be used by application programs.

Entry Control Blocks
ECB Format - continued

ECB page 2
- Used primarily by the control program.
- Holds data related to the entry that might impact system availability if damaged by the application program, such as control information that is used to manage the ECB and working storage blocks.
- Contains a field that points to the resource limit table (RLT), which contains the two levels of limits for each resource that is monitored.
- Contains a user-definable area.
- Uses a storage-access key of 2 for protection
  - The **PSW key must be explicitly changed in order to update this area.**

ECB page 3
- Used **only** by the z/TPF system control program
- Contains data and tables related to the management and activity of the entry

Entry Control Blocks
File storage (DASD) access

Although several different file storage devices (DASD), each with their own unique characteristics, are supported by the z/TPF system, all file accessing is conceptually done in a similar fashion.

The generic terms used for file accessing are find (read a record) and file (write a record).

A data level is one of the 16 pairs of data fields in page one of the ECB:
- File address reference word (FARW)
- Core block reference word (CBRW)

WAITC is an important macro used with I/O operations. WAITC is an application program request to delay further processing until all the pending I/O operations for the application program issuing the WAITC are completed. An I/O counter within the ECB is used for this control.

Entry Control Blocks
File storage (DASD) access - continued

The file address reference word (FARW) is used to pass a file address between the application and the control program.

The core block reference word (CBRW) is used to pass an address of a main storage block used for storing data. A main storage block is automatically obtained by the control program for a find macro request.

The allocation of data to physical file storage and the use of find and file macros are important:

- The record sizes used with find and file macros correspond to the same sizes used by the main storage allocation functions.
- Predefined records, such as indexes to more dynamic data, are managed in a file area known as fixed-file records. A set of data within the fixed-file area is identified as a record type. Within a fixed-record type each record is identified with an ordinal number.
- The dynamic requirements for the storage are satisfied by areas known as pools. The allocation of the pool record area to physical devices is also done during system generation. However, the pool record area represents a large repository of file storage accessible by all applications.

Entry Control Blocks
File storage (DASD) access - continued

The z/TPF system handles the details of obtaining the physical location of data, that is, addresses that are directly utilized by the hardware. This is done in stages:

- An application request to find or file a record must be preceded by one of two functions:
  - Use of the get file storage macro to obtain the address reference of a pool record.
  - Use of a system program to obtain the address reference of a fixed record, given the record type and ordinal number.
- An application program, in turn, passes a file address reference to I/O service routines through the use of FIND and FILE macros and the FARW of the ECB.

The file address information is in a format for either pool or fixed data record references by the time a find or file macro request is issued.

The use of data within fixed records as indexes to pool records is a common technique used in the z/TPF system environment.

Entry Control Blocks
Accessing and creating the ECB

The address of the ECB is in the ECB register which is part of the interface when control is passed from the control program to the application.

By convention of the z/TPF system:
- **Assembler** programs address the ECB using **register 9**.
- **C/C++ programs** use the function **ecbptr()** to obtain the ECB address.

OPZERO (Operation program Zero) initializes the ECB register when the ECB is created.

OPZERO creates and initializes an ECB with data.
- Where the input entry came ‘from’
- Where to send the response
- The input message itself
- And more….

Entry Control Blocks
ZSTAT - Display system status

Display status information about the z/TPF system, such as the number of allocated and available main storage blocks and the number of active ECBs.

Generate time-initiated displays of the status information.

Display information about I-stream utilization and I-stream-unique work list counts

Display information about the use of physical blocks based on owner name.

Display information about dumps.

Display information about the use of 31- , 64-bit, and recoverable system heap storage.

Display information about LPARs in your processor.

Entry Control Blocks
ZSTAT Output

```
STAT0041I 11.11.29 SYSTEM STATUS DISPLAY
                 RLOG     IOB    FRAME  COMMON     SWB     XWB     ECB  FRM1MB
ALLOCATED       17808    8192     2000     500   30720   30208    5000    6000
AVAILABLE       18707    8175     1989     497   29517   29425    4717    3437

ACTIVE ECBS               282
DLY/DFR ECBS                1
PROCESSED               49918
LOW SPEED                   0
ROUTED                      0
CREATED              78811794
SNA                         0
THREADS CREATED           252
TCP/IP INPUT                    518 763 262
TCP/IP OUTPUT                   540 887 621
END OF DISPLAY+
```

System Characteristics Agenda:

Entry Control Blocks

Message Processing

Message Flow

Memory Configuration

Main Storage Management

Message Processing
Overview

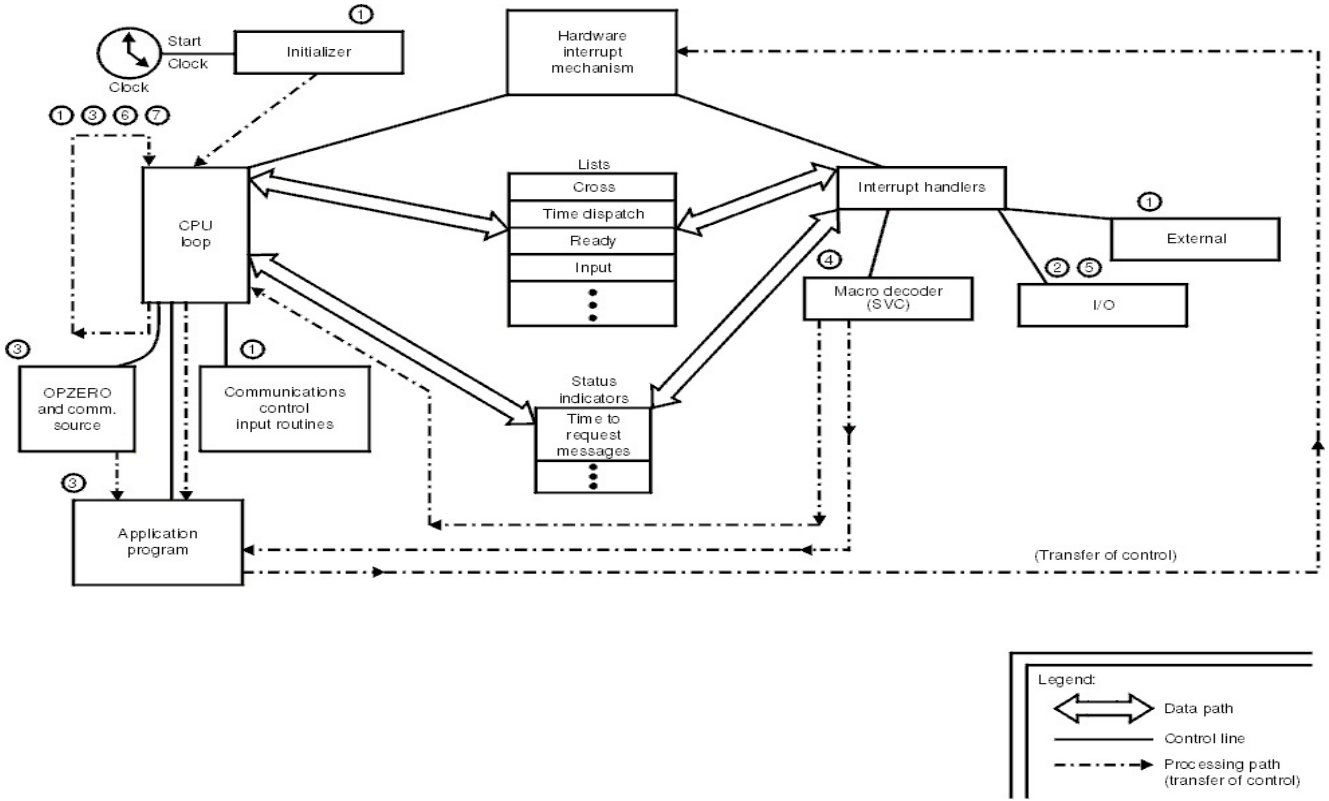
An input message causes an Entry (ECB) to be created

Application segments are dispatched for processing the input message.

The ECB makes requests of the control program for system services such as:
- Input and output (I/O)
- Storage allocation (both file and main storage)
- Program fetches (called Enter/Back in the z/TPF system)
- Release of control.

Many of the system services requested by an Entry have other system services implied. For example, if application program segment needs to read a file record, and the record is not already in main storage, then the system must:
- Obtain a working storage block in main storage for the record.
- Issue the I/O commands necessary to retrieve the record from file storage.
- Try to find another entry that can be started or continued during the delay caused by the I/O.

Message Processing
System Execution Overview



System Characteristics Agenda:

Entry Control Blocks

Message Processing

Message Flow

Memory Configuration

Main Storage Management

Message Flow
Message flow through z/TPF

The system is initialized at IPL and the CPU loop is in control.

The CPU loop keeps looping and checking for work on the cross, ready, input and deferred lists.

The CPU loop handles input messages
- invokes the communications control input routine which starts an I/O operation to read the data from the communication network.
- Messages arrive from actions initiated by the:
  - CPU loop
  - I/O interrupt handler.

The CPU loop finds an item on the input list. OPZERO creates and initializes an ECB, and control is routed to the application.

Message Flow
Message flow through z/TPF - continued

Fetch the application program from file using I/O.

The program segment is read into main storage and the program is started.

Running applications. An ECB obtains a block of main storage in which the output message is built.

Sending the reply. When processing of the input message is complete, the output message is transmitted to the end user over communication facilities.

Release resources and clean up. When all the processing required by an entry is complete, an EXITC macro is issued. All main storage utilized for processing the input message is released.

Message Flow
Entry termination (EXIT processing)

An exit macro request (EXITC) is used to stop processing an ECB in the z/TPF system.
- System resources held by the exiting ECB are returned to the system for use by other Entries (ECBs).
- Control is transferred to the exit program either because an exit macro was issued or because of the occurrence of a system error.

The exit program accomplishes the following:
- Disconnects from the ECB all programs associated with the ECB.
- Releases all main storage blocks, identified in CBRWs, used by the ECB.
- Releases all data records held by the ECB.
- Closes any tapes or unit record devices opened and assigned to the ECB at the time the EXITC was issued
- Ensures that all I/O initiated by the ECB is completed
- Releases the ECB.

System Characteristics Agenda:

Entry Control Blocks

Message Processing

Message Flow

Memory Configuration

Main **Storage** Management

Memory Configuration

Memory configurations allow a z/TPF image to run on different systems with differing amounts of available memory.

- Other non-memory system settings do not have to be duplicated and kept in sync in multiple locations
- Allows the same system definition to be used on a production system with large amounts of memory, and on a test system with much smaller amounts of memory

A memory configuration defines the allocation of core resources such as IOBs, SWBs, and frames.

- Memory allocations are defined in Keypoint A

As many as eight memory configurations are available to be used.

When the z/TPF system is IPLed, the control program matches a memory configuration to a physical processor by *best fit.*

- It uses the memory configuration that requires the most amount of storage after meeting the requirements of a minimum amount of both VFA and the 31-bit system heap.
- This ensures that as many system resources as possible are defined on a particular machine

The ZCTKA PREFER command can specify that a certain memory configuration is preferred so that the system will try to use this configuration first.

System Characteristics Agenda:

Entry Control Blocks

Message Processing

Message Flow

Memory Configuration

Main Storage Management

Main Storage Management
Virtual address space

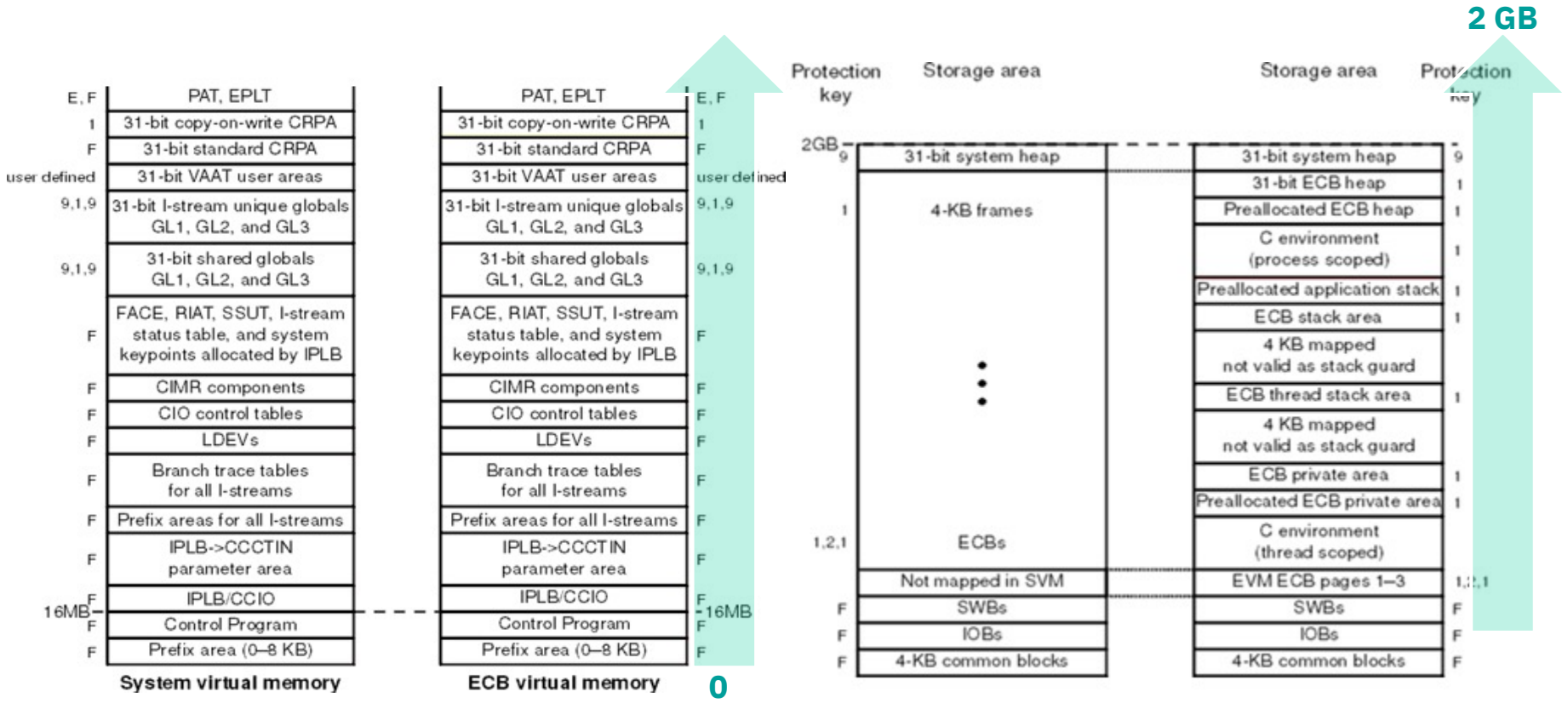
Virtual Storage
- Uses Dynamic Address Translation (a hardware feature) to translate a virtual address to a real address. DAT indicator is bit 5 in the PSW.
- Protects real storage from being corrupted.

The z/TPF system defines types of virtual address spaces, which are mappings of main storage.
- The system virtual memory (SVM) mapping is all of main storage.
- The ECB virtual memory (EVM) mapping is all of main storage that can be used (addressed) by an entry.

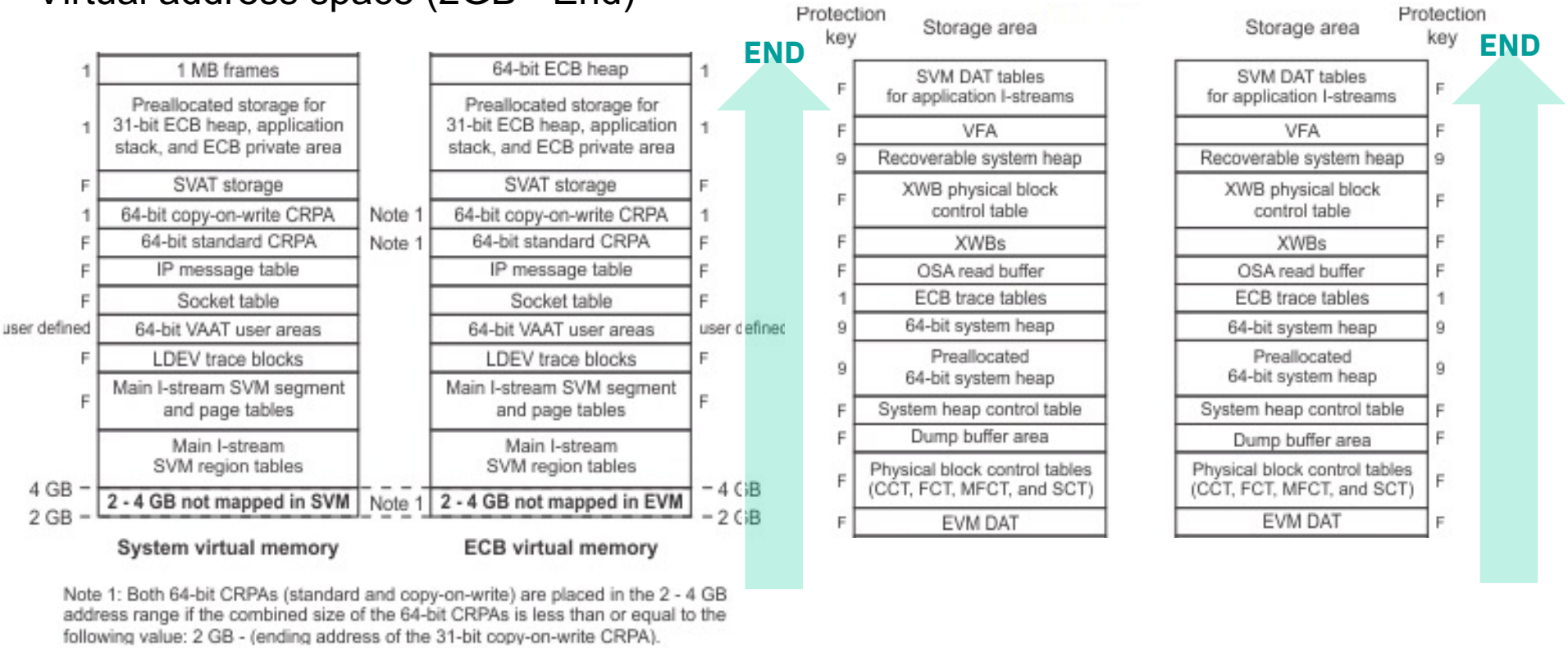
An application program (ECB-controlled program) uses the EVM mapping of main storage when processing.

The control program uses both the SVM and EVM mappings of main storage depending on the type of service it provides at any given time.

Main Storage Management
Virtual address space (0 to 2GB)



Main Storage Management
Virtual address space (2GB - End)



Note 1: Both 64-bit CRPAs (standard and copy-on-write) are placed in the 2 - 4 GB address range if the combined size of the 64-bit CRPAs is less than or equal to the following value: 2 GB - (ending address of the 31-bit copy-on-write CRPA).

Main Storage Management
Main, fixed and working storage

Main storage
– Consists of fixed storage and working storage.

Fixed storage
– Areas of main storage whose sizes are determined at system generation (i.e., system data records and tables). Fixed storage is permanently allocated by the control program.

Working storage
– Areas of main storage that are:
  - Available to application programs as system resources
  - The system control blocks used for managing an entry.

Main Storage minus fixed storage = Working Storage

The amount of working storage available to application programs depends on the number of system control blocks, common blocks, and other system resources in use at any given time.

Main Storage Management
Working Storage

Working storage is divided into distinct block types:
- 1M-byte frames
- I/O blocks (IOBs)
- System work blocks (SWBs)
- Extended system work blocks (XWBs)
- Entry control blocks (ECBs)
- 4K-byte frames
- 4K-byte common blocks.

These block types are physical storage blocks
- defined by owner information
  - a 32-byte descriptor, divided into three sections, that is saved in the associated control table entry for the storage block.
  - An owner could be application like ‘end transaction’ or more system oriented like ‘tape’
- Most macros that get storage by using physical blocks (such as GETBC, GSWBC, GCOMC, etc.) have an OWNER parameter. The owner name for an ECB is set by using the EOWNRC macro or the tpf_eownrc C function.

Main Storage Management
1M-byte frames and 4K-byte frames

1M-byte frames

– A 1M-byte frame is a block of virtual storage which is backed up by real storage. The 1M-byte frame control table (MFCT) manages all in-use and available 1M-byte frames. **There is no direct application interface.** It is only used by ECB heap, system heap and application stack storage when the pre-allocated areas are depleted. It is also used for 64-bit ECB heap requests.

4K-byte frames

– Used by the control program to satisfy requests for storage in a **standard block size (**128, 381, 1055, or 4K). The control program manages the allocation of these frames.
  - A 4K frame is a block of virtual storage. When a 4K frame is dispensed, the entire block is dispensed. This is for items that must have an address below the 2GB in real.
  - A 4K common frame is a block of virtual storage. When an entry requests a common block, it is allocated from this physical block regardless of its size.

Main Storage Management
4K-byte common frames

A 4K common frame is a block of virtual storage. When an entry requests a common block, it is allocated from this physical block regardless of its size.

Common blocks are available in standard block sizes.

The virtual addresses of common blocks are **mapped in each ECB's address space and backed by the same real address range**. This means that one ECB can "see" all common blocks just like all other common blocks. As a result, **common blocks are a form of shared storage.**

Common blocks are typically a limited resource and should be used with discretion.

Main Storage Management
Extended system work blocks (XWBs)

An extended system work block is a newer z/TPF block type. This block is used by system services such that you should reduce your SWB and 4 KB frame usage.

Characteristics
- Located above 4 GB in virtual storage.
- 4 KB in size, included in memory configurations (CORREQ / ZCTKA)
- Key F (like SWBs, IOBs, Common blocks)
- Syntax: USE=XWB on GSWBC and GETBC macros

System Usage
- Tape I/O queueing
  - Replaces SWBs for queueing control blocks.
  - Replaces 4 KB frames for data blocks.

- Tape logging and exception recording
  - Replaces 4 KB frames for data blocks.

- Work list expansion
  - Expansion of cross, ready, input, deferred.
  - Replaces common block to do the expansion.

Main Storage Management
Extended system work blocks

System Usage (continued)
- ECB create APIs
  - CREMC / CREDC / CREXC / CREEC / CRESC / SWISC CREATE / tpf_fork()
  - Replaces SWBs that are used to schedule requests.
  - Replaces 4 KB frames for when a core block is passed.

- Copy on write
  - Replaces 4 KB frame usage.

- ECB private
  - Used when the preallocated private area cannot fulfill a request
  - Replaces 4 KB frame usage.

Displays and monitor
- XWB usage tracked in the ECB resource monitor (ZECBM)
- Name-value pair collection.
- Resource collection by owner name and by limit set.
- ZSTAT, ZSTAT OWNER, ZDCLV ZACLV, ZSYSL

Main Storage Management
Extended system work blocks

Consider allocating a large number of XWBs

- More allocated XWBs allows you to improve toleration of tape performance anomalies.
- Example: 10,000 4 K tape data writes / second
    - 2 XWBs used per data write
    - Allocate 1,200,000 XWBs to queue for 1 minute
    - 1,200,000 XWBs use 5.15 GB

High volume SWB usage might continue to exist

- MQ is a significant user of SWBs. Since several users of SWBs were changed to use XWBs, MQ has less competition for this resource.
- MPIF/IPC support also uses SWBs and 4 KB frames.

Constraint relief for memory below 2 GB

- Possible to greatly reduce 4 KB frame allocation.

Main Storage Management
Logical storage blocks

Logical Storage Blocks
- Used for data records, message blocks, keypoint records, and program segments.
- These blocks can be private to an ECB or shared between ECBs depending on whether they are allocated from a 4K frame or from a 4K common frame. When logical storage blocks are located in an area private to the ECB (a 4K frame) they are private logical blocks or just logical blocks. When logical storage blocks are located in 4K common frames, they are referred to as common blocks and can be used to pass information among ECBs.

The relationship of a frame to (private) logical blocks is:
- One 4K frame = one 4KB logical block

The relationship of a common frame to common blocks is:
- One 4K common frame = one 4KB common block

When an application program requests a logical storage block, the control program service routine puts the block address into one of the 16 slots in the ECB called core block reference words (CBRWs).

Each CBRW is associated with a data level identified in macros as D*n* (for data level *n*) where *n* is a hexadecimal number between X'0' and X'F'. For example, D3 is known as data level three.

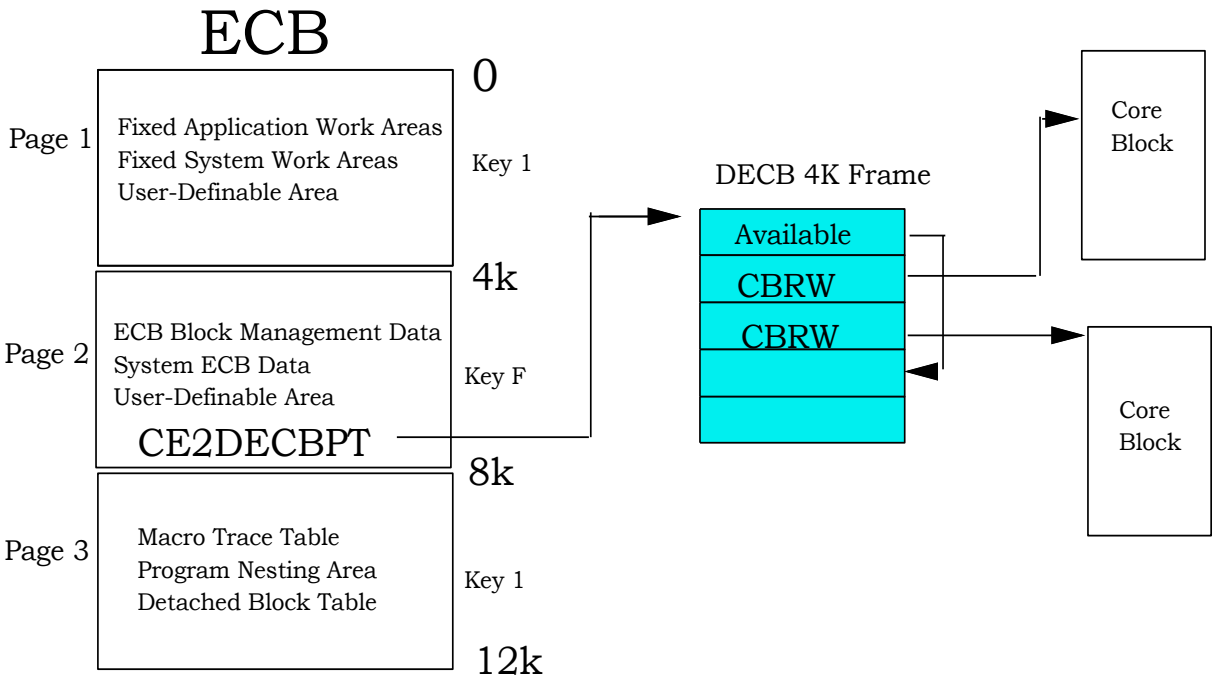
Main Storage Management
Data event control block (DECB) overview

Before DECB support, the z/TPF system restricted the number of entry control block (ECB) data levels (D0-DF) that were available for use to 16 (the number of data levels defined in the ECB). A DECB can be used in place of data levels for FIND/FILE-type I/O requests by applications.

Although a DECB does not physically reside in an ECB, it contains the same information as standard data levels:

- Core Block Reference Word (CBRW)
- File Address Reference Word (FARW)
- File Address Extension Words (FAXWs)
- Detailed error indicator.

Main Storage Management
Entry control block and DECB



Main Storage Management
ECB Resource Monitor (ecbRM)

The ECB resource monitor provides a centralized facility that monitors the use of selected system resources for each ECB.

The ECB resource monitor detects and, optionally, stops an ECB that requests excessive amounts of monitored resources.

- Enables the customer to monitor the use of system resources before the system has a problem, thus avoiding unexpected downtime, and ensuring system stability.
- Provides operational control and flexibility with the ZECBM command to turn the ECB resource monitor on or off.

The ECBRM detects and terminates any ECB requesting excessive quantities of monitored resources.

Two levels of resource limits exist for each monitored resource.

ecbRM may be controlled by the operator or by a program.

There are ecbRM related User Exits to modify what and how monitoring is done.

Learn more at: ECB Resource Monitor - IBM Knowledge Center 2020

Main Storage Management

ZECBM D

```
User:   ZECBM DISPLAY

System: ECBM0062I 13.23.26 ECB RESOURCE LIMIT TABLE DISPLAY FOR CORE COPY
                           FOR LSETNAME DEFAULT  SINGLE ECB MONITORING

                 1ST LIMIT    2ND LIMIT
        RESOURCE   WARNING      PERCENTAGE
        CMBK             0            0
        CRET            25          100
        FILE          1000          100
        FIND         50000           50
        GFSL             0            0
        GFSS             0            0
        GRFS             0            0
        RELF             0            0
        ROUT             0            0
        SERR             0            0
        SNAP             0            0
        SWBK             0            0
        SYSH             0            0
        TOUR             0            0
        TWRT             0            0
        XWBK             0            0
        FILT=500 FILR=50 FILIOB=25 FILSWB=0
        SETI=1000 SERA=50
        SNTI=0 SNRA=0
        WTOT=500 WTOR=50
        ecbRM MONITORING IS: ON
        1ST LIMIT ACTION IS: MESSAGE
        2ND LIMIT ACTION IS: MESSAGE
        END OF DISPLAY
```

Main Storage Management
ECB private area (EPA)

The get core macro (GETCC) dispenses a logical storage block in a standard size (381, 1055, or 4K). When the entry requests a (private) logical block, it is dispensed from the ECB private area, whereas when the entry requests a common block, it is dispensed from a 4K common frame.

The size of the ECB private area is defined with the CORREQ macro during SIP processing. You can change the size online by entering the ZCTKA ALTER command.

- ZCTKA DISPLAY can be used to display the size of the ECB private area.
  - The row containing EPRIV shows the size.
- ZCTKA ALTER EPRIV can be used to change the size of the ECB private area.
  - An IPL is required after the ZCTKA ALTER command in order for the change to take effect.
- SIP macro CORREQ EPRIV can be used to set the size of the ECB private area.

The ECB private area is private to an entry, it’s not accessible by other entries. Regardless of its actual location in main storage, an entry views its EPA as if it is located below the 2-GB bar.

Main Storage Management
ECB heap, application stack, and system heap

ECB heap
- Applications can request contiguous storage from the ECB heap by issuing a MALOC, CALOC, or RALOC macro, or by issuing a calloc, malloc, realloc, calloc64, malloc64, or realloc64 function. ECB heap storage is private to the ECB.

Application stack
- Application stack storage is acquired by z/TPF and is private to the ECB. This is a work area for things like register save areas, etc., for C and assembler programs.

System heap
- The GSYSC macro (or gsysc or tpf_gsysc function) obtains a variable-sized contiguous storage area from a system heap area. The system heap is not private to the ECB. System heap is requested in 4-KB or 1-MB units.
- Unlike ECB heap storage, system heap storage is not returned to the system when the ECB exits. System heap can be viewed as persistent storage. In this case, system heap storage is persistent across ECBs but temporary from the system point of view in that it can be released by the RSYSC macro (or rsysc or tpf_rsysc function) or the data is lost when the system IPLs.

Main Storage Management
31-bit, 64-bit, and recoverable system heap

31-bit system heap
- Located below the 2-GB bar. A minimum size is defined for 31-bit system heap storage. z/TPF might allocate more than the minimum size, depending on where the 4-KB frames end in the SVM or where the 31-bit ECB heap ends in the EVM, whichever is greater. This is then moved up to the next 1-MB boundary and from that point to the 2-GB bar is the size of the 31-bit system heap. All virtual addresses in this area are backed using 1-MB frames as GSYSC macro requests are processed.

64-bit system heap
- Located above the 2-GB bar. The size of the 64-bit system heap is determined by the total size of 1-MB frame storage. Applications can request contiguous storage from the ECB heap by issuing a GSYSC with HEAP=64BIT specified. System heap is not private to an ECB. All ECBs can see system heap. Preallocated used first. Backed by 1MB frames when needed.

Recoverable system heap (RSH)
- Located above the 2-GB bar. A fully preallocated area. Data is recovered on most IPLs. General rule: when VFA is not recovered, recoverable system heap will also not be recovered.
- Use the SHEAPC or tpf_sheapc() API to obtain and manage this type of heap.
- An IPL may not clean up fragmentation in this heap area.
- Orphaned buffers can impede fully using the heap area. What is an orphaned buffer? A block of heap that was allocated, but no finds (via SHEAPC) are executed to locate the buffer. See the ZMRSH command for options related to orphaned buffers as well as other RSH management options.

Main Storage Management
Copy-On-Write facility

Some programs may require special processing for the data in storage
- Programs that update static data
- Programs that have global constructors

These programs are placed in the copy-on-write core-resident-program area (CRPA). When the z/TPF system fetches this type of program, the program is page-protected in the ECB virtual memory (EVM) address space for all ECBs to avoid accidental corruption.

If a program attempts to update static data, the attempt to update the page-protected storage causes the system to assign a new “private” page to the ECB address space. The contents of the protected page are then copied into the private page.

Similarly, for global constructors, the indicator that the constructor has not been run is in page-protected storage. A new private copy of the page is made, and the private copy is updated to indicate the constructor has been run.

Thank You!

Questions or Comments?

Trademarks

Notes

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.