

Legal Notices

Any references to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

RACF*	DB2*	HiperSockets
Hiperspace	IBM*	IBM logo*
MVS	OS/390*	System/390*
System/360	Websphere*	VTAM*
z/Architecture	z/OS*	zSeries*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

UNIX is a registered trademark of The Open Group in the United States and other countries. Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

Other company, product and service names may be trademarks or service marks of others.



Agenda / Objectives

- The Performance Monitor has been presented to the User Group several times before
 - ▶ It is not my intention to repeat those presentations

- Describe what has been shipped
 - ▶ And how to migrate to it

- Highlight the differences in the IBM implementation

- Look at what could come next



Deliverables – Part 1

- The Performance Monitor was implemented in ALCS in two parts
- 1. The Monitor part AK16347 / UK15115
 - ▶ DXCPERF
 - ▶ PM0TB - Performance Monitor Table
 - ▶ APIDC (but not documented)
 - ▶ Dynamic TCB facility
 - TC0TB – TCB Control Table
 - CLCCC - CPU Loop Control Interface Macro
 - ▶ Many updates to monitor segments and ALCS ECB programs
- Designed to coexist with Jorge's ECB programs
 - ▶ Removed the need for Monitor CSECT user modifications
 - ▶ SCTGEN parameters: PERFMON and DYNTCB
 - ▶ ZCTCB command

Deliverables – Part 2

- 2. The ECB part – AK33864/UK24579, AK34228/UK24599 (IPARS)
 - ▶ PRFCC (replaces ZAPMC and RGTAG)
 - ▶ CPM1/2/3/4/5/6/7/8 (replace E9nn programs)
 - ▶ APF1
 - ▶ ZPERF command
 - ▶ Document APIDC
 - APIDC code in part 1 for current users but not documented
 - ▶ Many updates to monitor segments and ALCS ECB programs

Deliverables – Part 2

- 2. The ECB part – AK33864/UK24579, AK34228/UK24599 (IPARS)
- Installation for those who have not installed Jorge's code
 - ▶ Install the APAR fixes
- Further actions required in order to use the Performance Monitor
 - ▶ Optionally code and install programs APF1 and APF2
 - ▶ Generate a new SCTGEN with the PERFMON parameter
 - ▶ Restart ALCS

Deliverables – Part 2

- Migration from Jorge's Performance Monitor (JPM)
 - ▶ Remove any JPM updates in ECB-controlled programs
 - ▶ Remove the JPM ECB-controlled programs and macros
 - ▶ Install the APAR fixes
 - ▶ Optionally code and install programs APF1 and APF2
 - ▶ Modify the sample program CPM0
 - ▶ Restart ALCS
 - ▶ Run CPM0
 - Discards or Converts the history files to IBM format



Deliverables – Part 2

- Documentation PERFMON.pdf on the ALCS website
 - ▶ 1.0 ALCS Performance Monitor (New appendix to I & C Manual)
 - ▶ 2.0 ZPERF Command
 - ▶ 3.0 ZSTAT Command
 - ▶ 4.0 APIDC macro
 - ▶ 5.0 Customization (ECB Controlled exit - APF1/2)
 - ▶ 6.0 Messages and Codes

IBM Implementation – Part 2

- Restructured Tables
 - L4 → 2 x L3
 - System Table and Application Table
 - Includes areas for L5 – L8 pools plus spare
 - But this reduced the number of application slots from 55 to 39
 - 53 user slots reduced to 37 (2 slots used for SYST and DFLT)
 - Added ability to have up to 4 overflow application table records
 - 152 user slots
 - Conversion program for current users – CPM0



IBM Implementation – Part 2

Enhancements to Jorge's Performance Monitor

- 3 Application Levels
 - ▶ Will use 4 slots (for SYST and new DFLT applications)
 - Leaving 35 slots for user definition in first record

- Add all ZSTAT (SL0IT) values to System Table
 - ▶ Ability to display historic values
 - ▶ ZSTAT DATE=hhddmmyy | ddmmyy | mmyy

- 3 Performance Indicators



IBM Implementation – Part 2

- ... rest of Jorge's Performance Monitor
 - ▶ History File
 - Customer can choose to have History File or not
 - Index moved from L1 to #KPTRI(15) L2
 - Up to 5 years history (Jorge's implementation is fixed at 2 years)
 - Ability to release unwanted records
 - ▶ SLA → SPI - System Profile Indicator
 - CPU
 - APPL
 - I/O



CPM0 – Sample History File Conversion Program

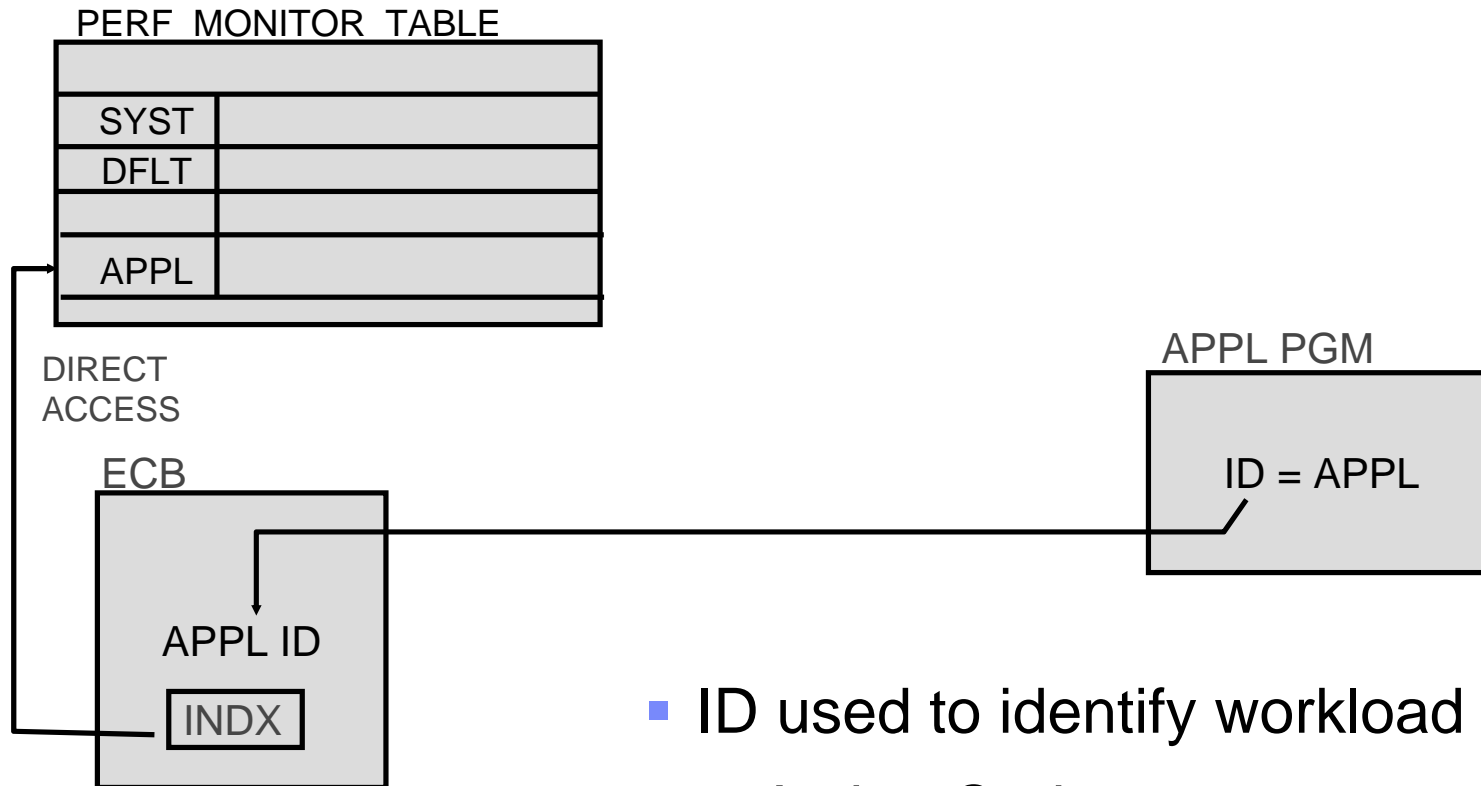
- Converts the Performance Monitor History File records
 - ▶ Alternatively can release the old history structures

- CPM0 must run when ALCS is restarted for the first time with an Enabled Performance Monitor
 - ▶ Any History collected after that restart will be lost

- CPM0 invoked by "ZDRIV CPM0" after you have deactivated the Performance Monitor

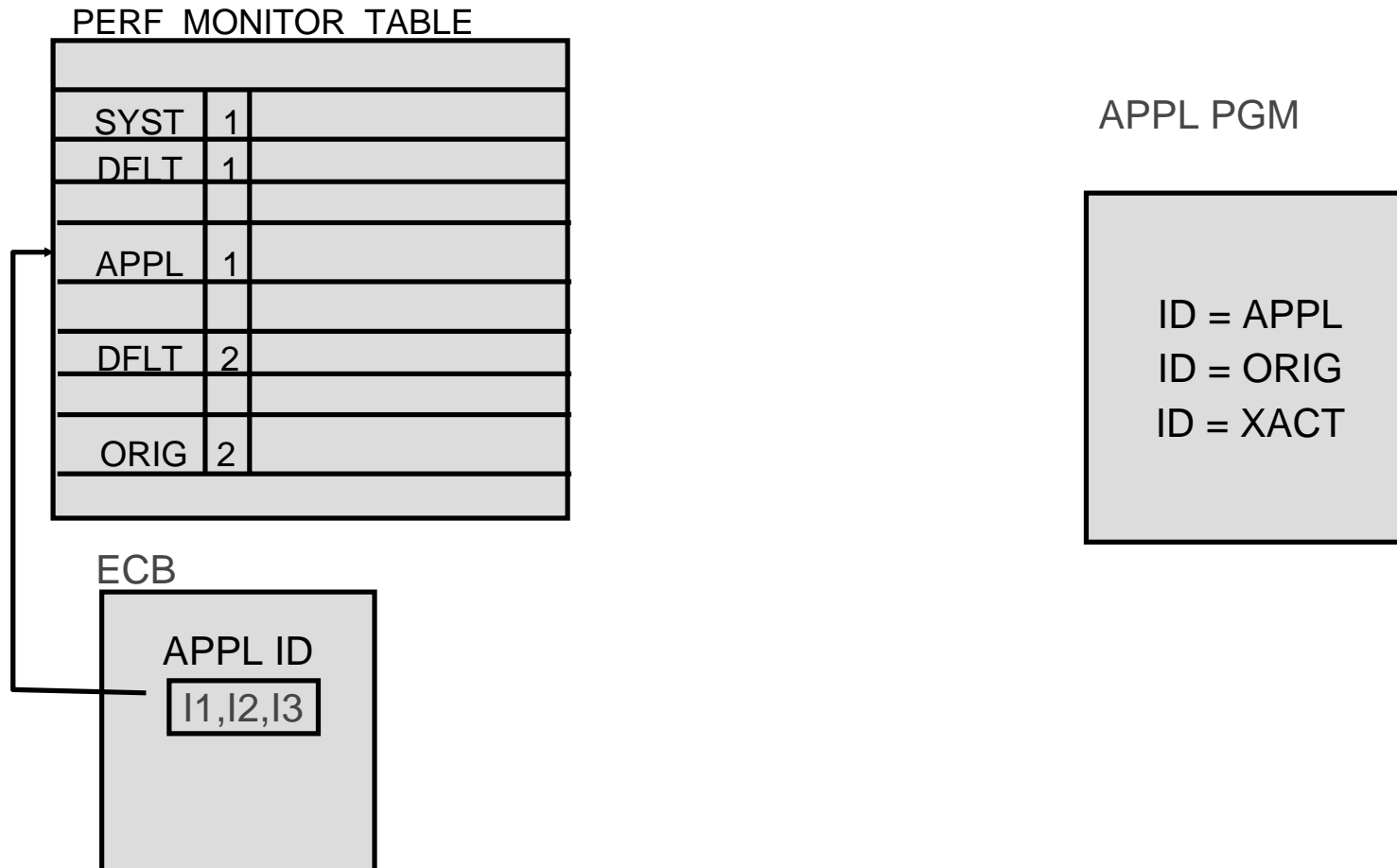
- Re-initialize and activate the Performance monitor after message
 - ▶ "CPM0 COMPLETED" IS DISPLAYED BY "ZPERF INIT CORE".

Application Identification - JPM



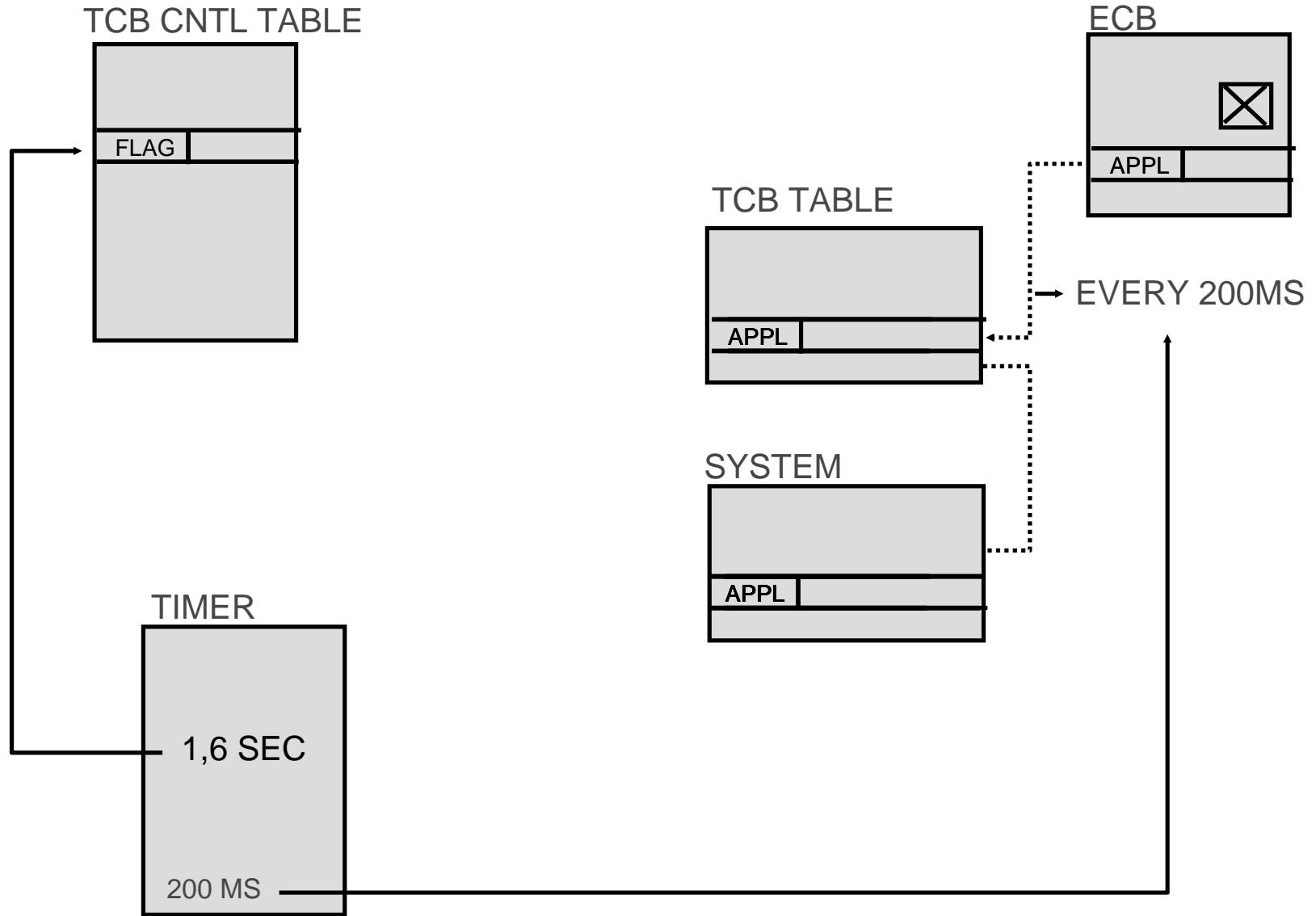
- ID used to identify workload eg:
 - ▶ Action Code
 - ▶ Application Suite
 - ▶ Message Origin

Application Identification – IBM implementation

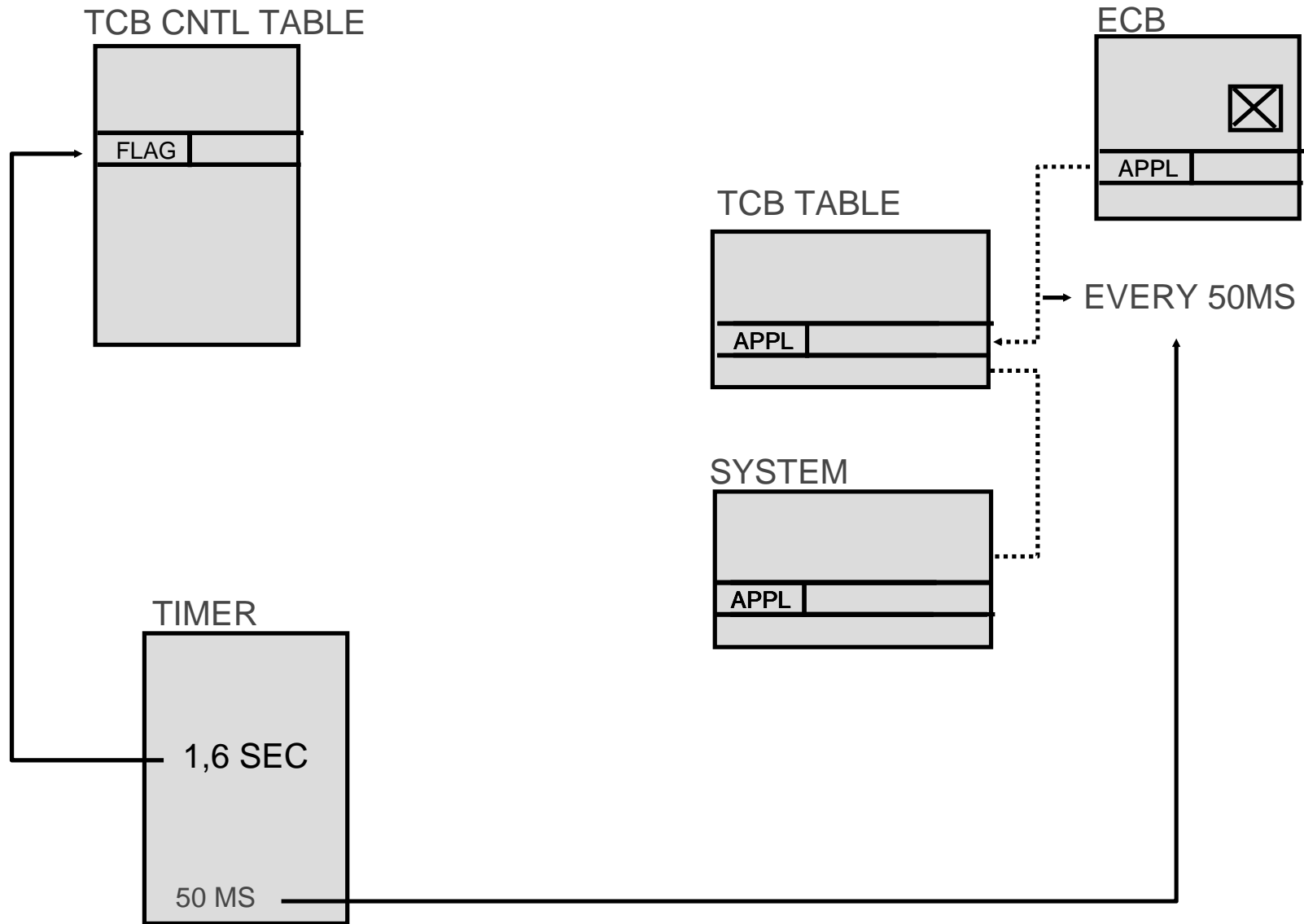


- 3 DIFFERENT APPL ID's MAY BE USED FOR ACCOUNTING SAME ENTRY

Collection Cycle - JPM

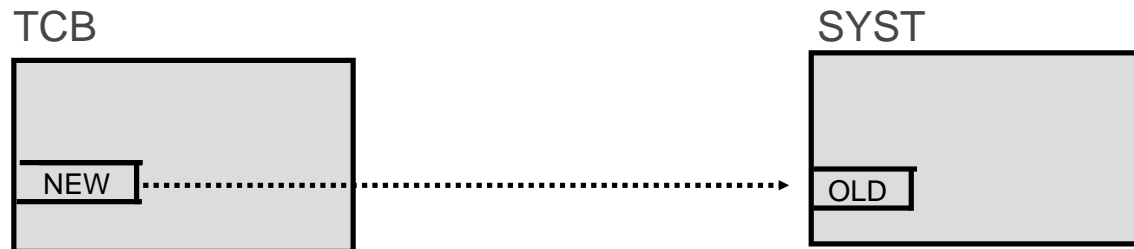


Collection Cycle – IBM implementation



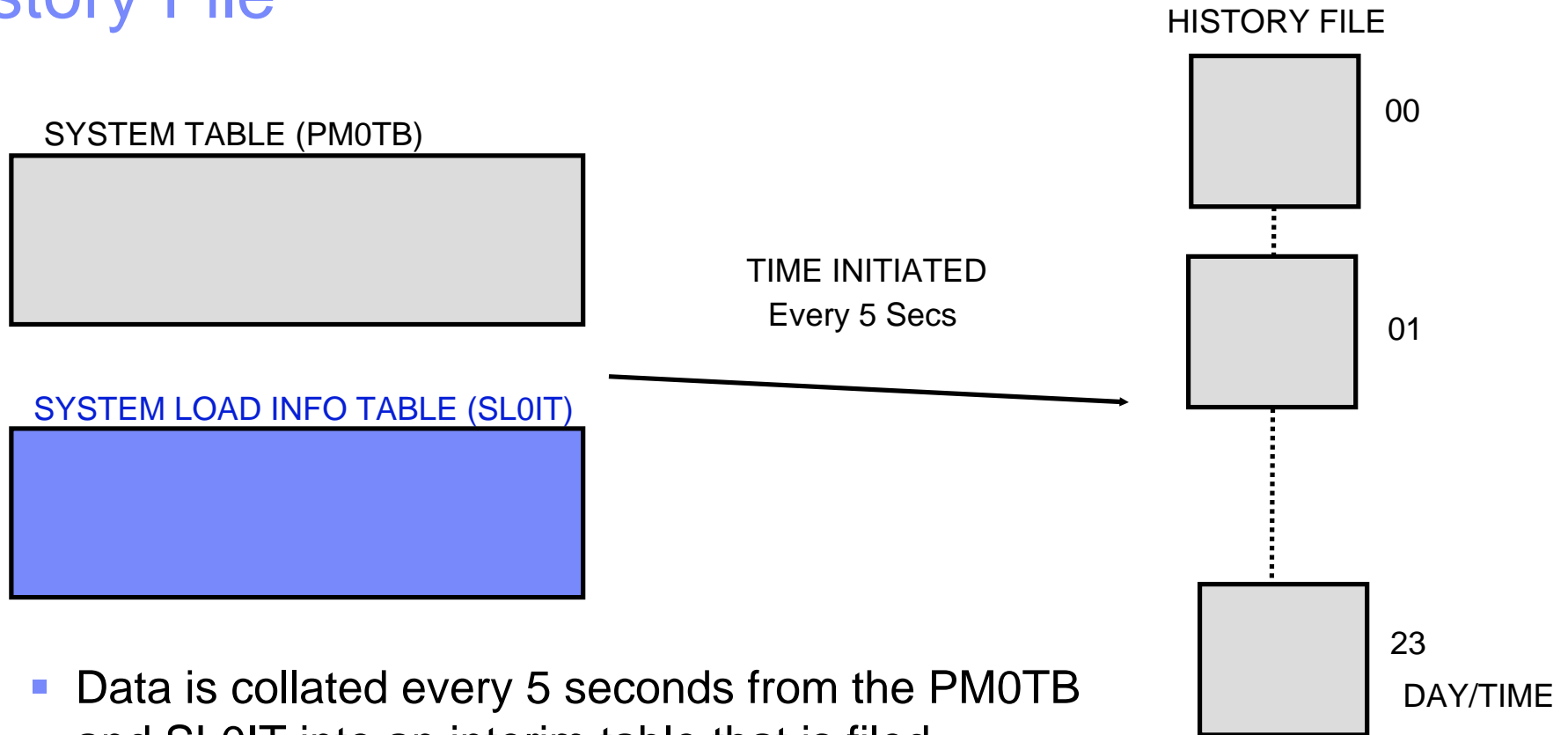
Collection Cycle Smoothing

- OLD VALUE - WEIGHT 15 (JPM was 4)
- NEW VALUE - WEIGHT 1



$$\text{SYST} = \frac{\text{NEW} * 1 + \text{OLD} * 15}{16}$$

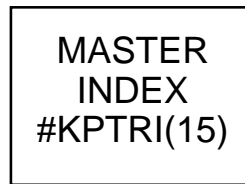
History File



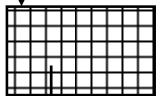
- Data is collated every 5 seconds from the PM0TB and SL0IT into an interim table that is filed
- This interim data is summed and divided by the number of interim recordings in an interval (1 hour) to derive the mean values that are recorded in the history file records.

History File

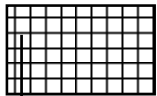
- Kept on File for up to 5 Years (JPM 2 Years)
 - Current period (12 months) can not be deleted



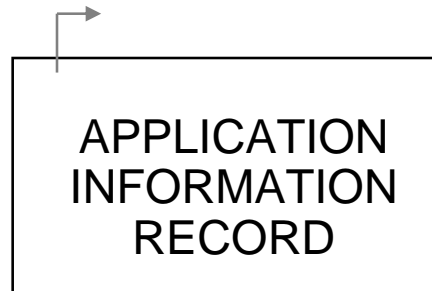
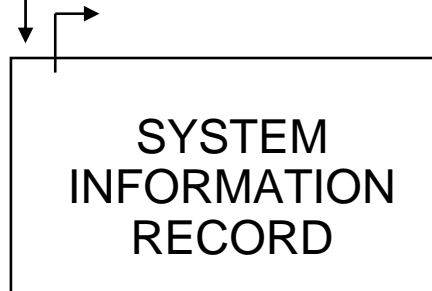
#KPTRI(15) contains an item for each month/year



L1 Pool record holds index for 31 day items and an average day item



L1 Pool record holds index for 24 hour items and an average hour item



L3 Pool Usage with one application record kept 5 years

$$2 \times 25 \times 32 \times 12 \times 5 = 96000$$

Test System (current period) requirement

$$2 \times 25 \times 32 \times 12 = 16000$$

System Profile Indicators – SPIs (Jorge’s SLAs)

- Application
- CPU
- I/O

- System I/O

- Message
 - ▶ Shows the input message rate per ECB

- VFA

System I/O SPI

- The System I/O performance indicator shows what percentage of ALCS I/O is not directly driven by the entries but by the system to perform background tasks and housekeeping.
- The value is:

$100 \times (\text{Total I/O} - \text{Sum of (Total ECB Reads + Total ECB Writes)})$

$\text{Sum of (Total ECB Reads + Total ECB Writes)}$



VFA SPI

- The VFA SPI takes the ECB collected figures for finds, reads, files and writes to indicate the efficiency of VFA

- The value is:

$$100 \times ((\text{ECB Reads} - \text{ECB VFA Reads}) + (\text{ECB Writes} - \text{ECB VFA Writes}))$$

$$\text{Sum of (ECB Reads + ECB Writes)}$$

ZPERF

- ZPERF CPU | I/O | MSG | POOL | SPI | SYS | TASK | VFA
 - ▶ Displays various performance reports

- ZPERF ACTIVATE / INACTIVATE
- ZPERF INIT FILE|CORE
- ZPERF ADD NAME = XXXX , LEVEL = 1 | 2 | 3
- ZPERF DEL NAME = XXXX
- ZPERF AVERAGE and ZPERF PURG
- ZPERF FORCE ECB = XXXX
 - ▶ Assign ECB to 'PERF' APPL
 - ▶ Dynamic Investigation of Batch Type Process

History update exit program – APF1

ALCS conditionally enters APF1 when updating the history records

- APF1 must return with the number of years (beyond the current year) for which history records will be kept online.

- There are no special entry conditions for APF1.
- Program APF1, and any program that APF1 calls, **must not**:
 - ▶ Modify EBW000 through EBW103 or EBX000 through EBX103
 - ▶ Use ECB storage levels
 - ▶ Modify registers R00 through R07
- APF1 must return using BACKC

- On return from APF1, register R15 must contain the number of years (0-5)

History average exit program – APF2

- ALCS conditionally enters APF2 during average processing
- APF2 must return with the addresses of two tables:
 - ▶ one of the hours to be used when creating the day average file record (R14)

Offset	Length	Contents
0	2	Number of entries in the table, a halfword value from 1 to 24
2	2	First hour value. Two numeric characters from 00 to 23
4	46	Up to 23 further hour values

- ▶ one the days to be used when creating the month average file record (R15)

Offset	Length	Contents
0	2	Number of entries in the table, a halfword value from 1 to 31
2	2	First day value. Two numeric characters from 01 to 31
4	60	Up to 30 further day values

- The default is to use all hours of a day and all days of a month
- Same entry and processing restrictions as APF1



Is there a need for Part 3 ?

- Performance Monitor identified 'batch-type' problem programs
 - ▶ LODIC used to throttle programs
 - ▶ Not satisfactory
 - LODIC ECBCREATE uses single check level for all appls
 - LODIC CONDITIONAL DEFER is fixed
 - ▶ Solution: Throttle Mechanism



Throttle Mechanism

- Macro service to control resources used by batch-like applications
- Throttle Control Table
- Macro THTRC results in ENTRC to Throttle control program
- 4 ECB controlled Throttle programs
 - ▶ Initialisation
 - ▶ Police
 - ▶ Command Interface
 - ▶ Macro Service
- Command ZZTHR



Throttle Table

- In core table read from file at restart
- Updated via Z command
 - ▶ ZZTHR ADD NAME=RECP, NLOOP=50
 - ▶ ZZTHR ALT NAME=RECP, NLOOP=5
 - ▶ ZZTHR THRS NAME=RECP, IOQ=5
 - ▶ ZZTHR DISP

		Normal Set			Threshold			Restrictive Set		
	State	nloop	wait1	wait2	cpu	i/o	user	nloop	wait1	wait2
RECP	Norm	50	0	0	0	5	0	5	0	0

Throttle Table

▶ THTRC LOAD

- Returns value 50

		Normal Set			Threshold			Restrictive Set		
	State	nloop	wait1	wait2	cpu	i/o	user	nloop	wait1	wait2
RECP	Norm	50	0	0		5		5	0	0

▶ SYSTEM LOAD INCREASES

- I/O queue time exceeds 5 ms
- Throttle police routine updates table

		Normal Set			Threshold			Restrictive Set		
	State	nloop	wait1	wait2	cpu	i/o	user	nloop	wait1	wait2
RECP	Restrict	50	0	0		5		5	0	0

▶ THTRC LOAD

- Returns value 5



Throttle Table

		Normal Set			Threshold			Restrictive Set		
	State	nloop	wait1	wait2	cpu	i/o	user	nloop	wait1	wait2
APPL	Norm	10	1	2	10	0	0	4	3	9

- ▶ THTRC WAIT
 - Application waits 1 second
 - After 10 THTRC WAIT macros application waits 2 seconds

- ▶ SYSTEM LOAD INCREASES
 - ALCS CPULOAD exceeds 10% of partition
 - Throttle police routine updates table

		Normal Set			Threshold			Restrictive Set		
	State	nloop	wait1	wait2	cpu	i/o	user	nloop	wait1	wait2
APPL	Restrict	10	1	2	10	0	0	4	3	9

- ▶ THTRC WAIT
 - Application waits 3 seconds
 - After 4 THTRC WAIT macros application waits 9 seconds