SCP Various Enhancements

CP Team

Michael Shershin, TPF Development Lab

IBM **z/TPF**

April 11, 2016

Agenda

PJ43653 and PI50297 – 1 millisecond time slice

PJ43266 – mymalloc support

PJ43067 – ECB stack validation

PJ43633 – ZDECB enhancements

PJ43632 – ZDHST enhancements

Futures – Tape redirect

PJ43653 and PI50297

1 millisecond time slice

1 millisecond time slice

- PJ43653 and PI50297 are on PUT 13
- Prior:
  - 10 milliseconds is minimum run time before ECB is time sliced
  - Could impact existing transactional traffic if significant number of ECBs use time slice
- Current:
  - 1 millisecond is minimum run time before ECB is time sliced

1 millisecond time slice changes

- CPU timer external interrupt changed to 1 millisecond
  - Time slice work done every millisecond
  - ECB time out checks (CTL-10) done every millisecond
  - Other work done by CPU timer external interrupt is on 10 millisecond boundary
- ZTMSL command ADD and CHANGE parameters:
  - RUNTIME option accepts minimum value of 1 millisecond
- TMSLC macro with ASSIGN parameter
  - RUNTIME option accepts minimum value of 1 millisecond

1 millisecond time slice changes

- TPFDF internal changes
  - TPFDF uses internal control fields for ECB time out checks (CTL-10)
  - Internal field (PFXATMR) changed
    - Now a count of 1 millisecond intervals
    - Was a count of 10 millisecond intervals
  - Internal DFDFRC and DFDLAY macros changed

Example

```
==> ZTMSL DISPLAY IBMHIPRI
CSMP0097I 13.42.30 CPU-B SS-BSS  SSU-HPN  IS-01
TMSL0003I 13.42.30
 TIME SLICE ATTRIBUTES FOR NAME IBMHIPRI ON FILE

    MAXECB-  50  MAXTIME-  10000  MINSUSP- 100  RUNTIME-100  SLICES-    0
 END OF DISPLAY+

==> ZTMSL CHANGE IBMHIPRI MINSUSP-1 RUNTIME-1
CSMP0097I 13.43.06 CPU-B SS-BSS  SSU-HPN  IS-01
TMSL0005I 13.43.06
 OLD TIME SLICE ATTRIBUTES FOR NAME IBMHIPRI ON FILE

    MAXECB-  50  MAXTIME-  10000  MINSUSP- 100  RUNTIME-100  SLICES-    0

 NEW TIME SLICE ATTRIBUTES FOR NAME IBMHIPRI ON FILE
 _
    MAXECB-  50  MAXTIME-  10000  MINSUSP-   1 RUNTIME-  1  SLICES-    0
 END OF DISPLAY+
```

Example

```
==> ZTMSL ADD IBMJAVA MAXECB-9999 MAXTIME-0 MINSUSP-1 RUNTIME-1
CSMP0097I 13.41.58 CPU-B SS-BSS  SSU-HPN  IS-01
TMSL0004I 13.41.58
 NEW TIME SLICE ATTRIBUTES FOR NAME IBMJAVA  ON FILE

    MAXECB-9999  MAXTIME-     0  MINSUSP-   1  RUNTIME-  1  SLICES-    0
 END OF DISPLAY+
```

PJ43266

mymalloc

mymalloc

- PJ43266 is on PUT 12
- Reduce instructions for ECB heap requests
  - Application that has a large number of in use ECB heap buffers of a similar size
  - Trade-off:
    - No malloc diagnostics (trace and obtaining program information)
    - Cannot tag a mymalloc buffer (with tpf_eheap_tag)
    - No checks for corrupted mymalloc heap at free time

ECB heap control entry

- Each malloc request obtains a ECB heap control entry
  - Great for diagnostics
  - Large number of ECB heap control entries becomes expensive
    - 150 ECB heap control entries exist when ECB is created
    - A 4K system heap chunk is obtained for every 31 additional ECB heap control entries
    - A 1meg system heap chunk is obtained when 970 ECB heap control entries are used
      - Holds the hash for ECB heap control entries

mymalloc buffer handling

- Obtain large ECB heap buffer
  - Distribute small fixed size buffers from the large buffer
  - Example: mymalloc for buffer of 8 bytes:
    - Do malloc to obtain one buffer to hold 512 buffers of 32 bytes (16,384 bytes)
    - Return one buffer of 32 bytes to mymalloc caller
    - Next mymalloc caller for 8 bytes gets another buffer of 32 bytes
    - Only one ECB heap control entry for the large (16,384 bytes) buffer
    - No ECB heap control entry for small buffer of 32 bytes

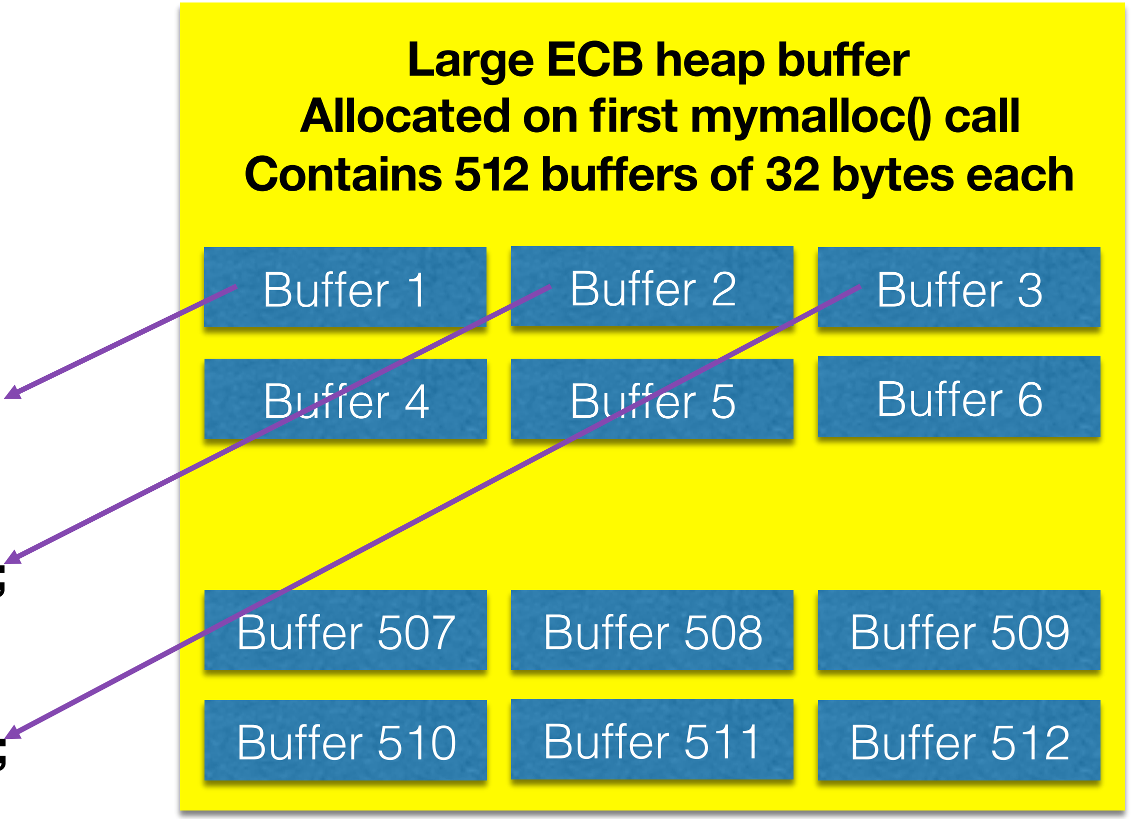
Example of buffer handling

```
void *workArea1;
void *workArea2;
void *workArea3;
int size;

size=8;
workArea1 = mymalloc(size);


workArea2 = mymalloc(size);


workArea3 = mymalloc(size);
```



mymalloc APIs

- C language
  - mymalloc(), mycalloc() – obtains ECB heap buffer
  - myfree() – returns a ECB heap buffer obtained with mymalloc
  - myrealloc() – re-sizes a ECB heap buffer obtained with mymalloc
- Assembler
  - MYMALOC, MYCALOC
  - MYFREEC
  - MYRALOC
- Must use myfree() or MYFREEC to return a mymalloc buffer
  - An error will be given is free() or FREEC is used

mymalloc buffers

- Three mymalloc buffer types
- Default buffer types:
  - Small: size requests of 1 byte to 32 bytes: 512 buffers allocated
  - Medium: size requests of 33 bytes to 64 bytes: 512 buffers allocated
  - Large: size requests of 65 bytes to 128 bytes: 128 buffers allocated
- User exit allows customization of mymalloc buffers
  - Function name: mymallocUserExit()
  - In file umymalloc.c
  - Customization can be unique per ECB

mymalloc

- mymalloc requests use standard malloc in the following conditions
  - mymalloc is disabled (ZSTRC ALTER NOMYMALLOC)
  - Threaded ECB
  - Heap check mode is active (ZSTRC ALTER HEAPCHECK)
  - Request size is not managed by mymalloc

mymalloc for C++

- Ability to use mymalloc for new and delete operators
  - In .mak file, add ARCHIVES statement to use mymalloc
    - new will use mymalloc()
    - delete will use myfree()

- Example taken from test driver qzz5.mak

```
APP := QZZ5
APP_ENTRY := QZZ5
APP_EXPORT := ENTRY
ARCHIVES := mymalloc
```

Example

Taken from rlch.asm:

```
*                                                                        @PJ31406
* Setup chain of file addresses being chased                             @PJ31406
*                                                                        @PJ31406
         LA    R5,RLCH_LEN                    Size of chaining item      @PJ31406
         MYMALOC SIZE=R5                      Get a chaining item        @PJ43266

...
... Much later in the program
...

RLCH22A1 DS    0H                                                        @PJ37297
         MYFREEC BLOCK=R7                                                @PJ43266
         DECBC FUNC=RELEASE,DECB=(R1)                                    @PJ31406
```

PJ43067

ECB stack validation

ECB stack validation

- PJ43067 is on PUT 12
- Several customers experienced ECB stack corruption
  - OPR-4 happens when data collection program collector is active
- Provide ability to identify ECB stack corruption
  - Validates addresses in backward chain field in ECB stack
    - Validates up to 100 back chain fields in stack
    - Stops when contents of back chain is zero (initial stack frame)
  - Validation done at the following times
    - Entry to C function and Exit from C function
    - ENTRC and BACKC

ECB stack validation error

- Address in back chain field (ICST_BCH) is:
  - Not zero and
  - Not within the virtual area for the ECB stack
- System error 064009 is taken
- ECB is exited

ECB stack validation controls

- Turn on: ➔ ZSTRC ALTER STACKVAL
- Turn off: ➔ ZSTRC ALTER NOSTACKVAL

ECB Stack validation

Traverse back chain

12F5F240 – current stack

- 12F5F800 – current back chain
- 12F5FC80 – next back chain
- 12F5FE40 – next back chain
- 0 – stop validation

Back chain address is valid when

Address is zero, or

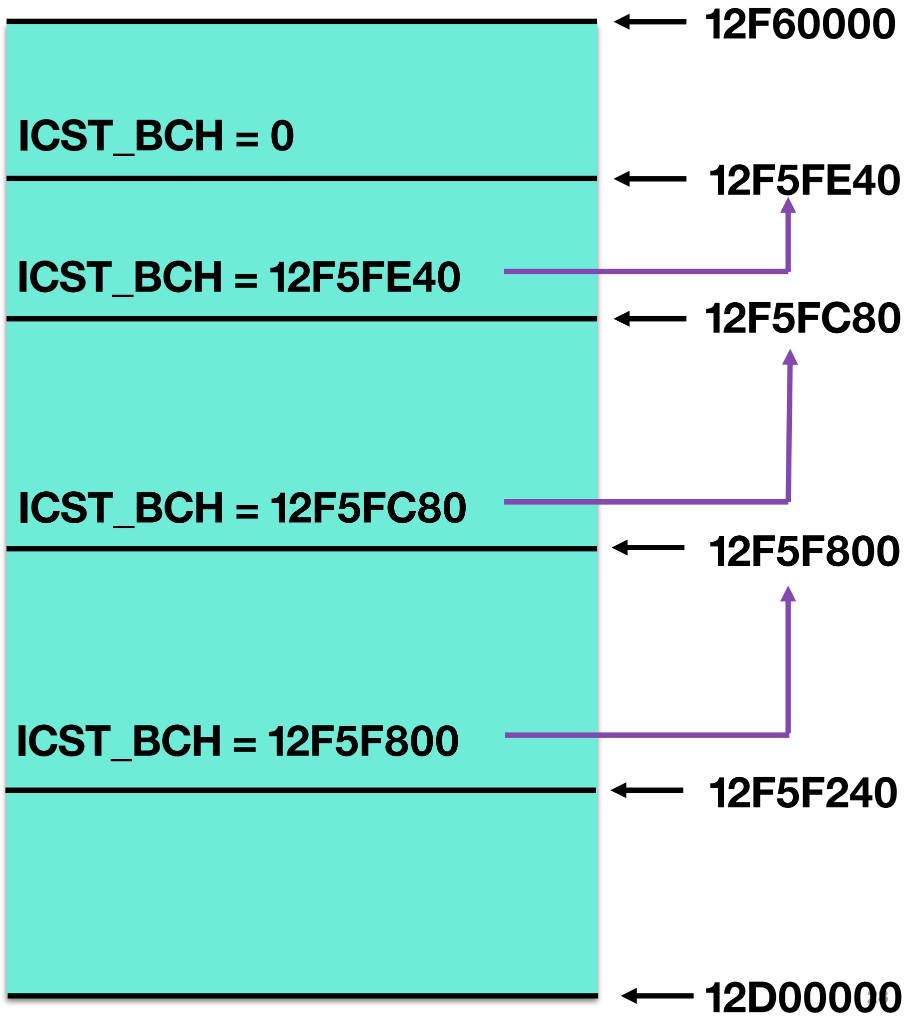
Address is within ECB stack virtual area

12D00000 < address < 12F60000

If address is not valid

System error 064009 is taken

ECB is exited



ECB stack validation recommendation

- Use in test systems
- Overhead will vary
    - Stack depth affects overhead
- May be able to use in production
    - Initially use in off hours and measure overhead
    - Idea: use during low traffic periods when application programs are loaded, activated, and used
        - Provides additional diagnostics if error happen

PJ43633

ZDECB enhancements

ZDECB - display in use ECBs

- PJ43633 is on PUT 12
- New option: USER
  - Summary display of in use ECBs that includes
    - ECB owner name
    - Named limit set (LSETNAME)
- New filter options for all ZDECB summary displays of in use ECBs
  - Selection by owner name: OWNER-*ownername*
    - Qualifier can be for high level owner name
    - Qualifier can be for high and mid level owner name
  - Selection by named limit set: LSETNAME-*lsetname*

Example

```
==> ZDECB USER 10 OWNER-INETD
CSMP0097I 14.34.20 CPU-B SS-BSS  SSU-HPN  IS-01
DECB0014I 14.34.20 DISPLAY ECB SUMMARY
ECB ADDR IS  PGM     TRC MIN SC LSETNAME OWNER NAME
1288E000  4 CLTW    CLTW 999 24 DEFAULT  INETD   MONITOR BSS
128A3000  3 COMX    COMX 999 24 DEFAULT  INETD   LISTENERFTP-BSS
128D3000  2 CLTZ    CLTZ 999 24 DEFAULT  INETD   LISTENERSYSLOGD-BSS
128FD000  4 COMX    COMX 999 24 DEFAULT  INETD   LISTENERTEST2-BSS
12912000  3 COMX    COMX 999 24 DEFAULT  INETD   LISTENERMATIPA-BSS
12918000  4 COMX    COMX 999 24 DEFAULT  INETD   LISTENERTFTP-BSS
1291E000  2 COMX    COMX 999 24 DEFAULT  INETD   LISTENERDNS-BSS
1292D000  2 COMX    COMX 999 24 DEFAULT  INETD   LISTENERZTPFSOAP-BSS
1293F000  2 COMX    COMX 999 24 DEFAULT  INETD   LISTENERTEST1-BSS
1295A000  3 COMX    COMX 999 24 DEFAULT  INETD   LISTENERMATIPB-BSS
TOTAL        40
END OF DISPLAY+
```

Example

```
==> ZDECB 0 OWNER-drvrDFCA
CSMP0097I 14.25.02 CPU-B SS-BSS  SSU-HPN  IS-01
DECB0014I 14.25.02 DISPLAY ECB SUMMARY
ECB ADDR  SSU IS  PGM     TRC MIN SC  ORIGIN   I H      DSP  SVC
1584F000 WP1   2 UTDF  * UBI5 999  6 CREM QDCH 1 1    41838 FINWC 34A30211 _
158A3000 WP1   1 UTDF  * UBI5 999  6 CREM QDCH 1 1    41838 FINWC 34A2FD45
158AF000 WP1   3 UTDF  * QDCI 999  6 CREM QDCH 1 1    41826 FIWHC 704B05AD
158BE000 WP1   2 UTDF  * UBI5 999  6 CREM QDCH 1 1    41838 FINWC 34A20ECD
15912000 WP1   4 UTDF  * UBI5 999  6 CREM QDCH 1 1    41838 FINWC 34A2F521
TOTAL          5
END OF DISPLAY+
```

Example

==> ZDECB STAT OWNER-drvrSOCK

```
CSMP0097I 14.40.47 CPU-B SS-BSS  SSU-HPN  IS-01
DECB0014I 14.40.47 DISPLAY ECB SUMMARY
ECB ADDR  SSU IS  PGM      TRC MIN SC    MILS    F4K F1MB    FIND      FILE      GETF
15681000 HPN   1 CTS7  * CTS7      30      2K      3    2       8         0         0  _
14AE7000 HPN   1 CTS4  * CTS4 999 51       9      2    1       1         0         0
14B50000 HPN   1 CTS4  * CTS4 999 51       8      2    1       1         0         0
14B0B000 HPN   1 CTS4  * CTS4 999 51       7      2    1       1         0         0
14B8C000 HPN   1 CTS4  * CTS4 999 51       7      2    1       1         0         0
TOTAL          5 _
END OF DISPLAY+
```

PJ43632

ZDHST enhancements

ZDHST - display dump history

- PJ43632 is on PUT 12
- New parameter: PAST-*hours*
  - Display dump information for the previous number of hours
  - Previously required a start date and time
- Filter parameters on DBA (dump buffer utilization) option
  - Start date / end date
  - PAST number of hours
  - Previously used all available data

Example

==> ZDHST DISPLAY TOTALS PAST-48

```
CSMP0097I 14.56.51 CPU-A SS-BSS  SSU-BSS  IS-16
DHST0007I 14.56.51 SYSTEM ERROR TOTALS DISPLAY
FILTERS:
    DISPLAY TOTALS PAST-48
TARGET SS: BSS     RETENTION: 5
PROC  TYPE        CTL      OPR      SNAP  MANUAL  TOTALS _
 A    DUMP          0        1        0       0       1
      NODUMP        0        5        1       0       6
END OF DISPLAY+
```

Example

==> ZDHST DISPLAY TYPE-OPR PAST-48

```
CSMP0097I 15.04.02 CPU-A SS-BSS  SSU-BSS  IS-16
DHST0005I 15.04.02 SYSTEM ERROR DETAILS DISPLAY
FILTERS:
    DISPLAY TYPE-OPR PAST-48
TARGET SS: BSS     RETENTION: 5
SE #   TYP  SYSERR    PROC IS  DATE/     SS/   PRGM   EBROUT/   TAPE _
                               TIME      SSU   TRACE  LOADSET
       OPR I00000004 A    08  05Mar16   BSS   CP     000000A
                              19:35:06  BSS   M597
       OPR I00000004 A    07  05Mar16   BSS   CPP1   000000A
                              21:21:48  BSS   M597   BASE     _
006815 OPR I00DB0138 A    13  06Mar16   BSS   UTDF   010000A   T1G766
                              13:38:18  BSS   CADB   UTDFDBG3
       OPR I00000004 A    03  07Mar16   BSS   CP     000000A
                              02:17:50  BSS   M597
       OPR I00000004 A    04  07Mar16   BSS   CP     000000A        _
                              04:57:34  BSS   M597
       OPR I00000004 A    12  07Mar16   BSS   CP     000000A
                              08:12:51  BSS   M597
END OF DISPLAY+
```

Example

==> ZDHST DISPLAY DBA PAST-48

```
CSMP0097I 15.16.28 CPU-A SS-BSS  SSU-BSS  IS-16
DHST0008I 15.16.28 DBA UTILIZATION DISPLAY
CURRENT DBA (MB) - 100
CURRENT PEAK THRESHOLD - 10%
    DATE-TIME           UTIL - MB   UTIL - %   CPU
20160306-13.38.18              25          24    A
END OF LIST+
```

Other enhancements delivered on PUT 12

Other enhancements – available now

- These enhancements were discussed at the last TPF Users Group
- PJ42459 – 2GB page support
  - Performance improvement for zEC12 and z13 machines
  - Uses one TLB entry for 2 GB of memory
- PJ43353 – Format 1 Global enhancements
  - Allows I-stream growth by reducing I-stream unique storage areas below 2 GB
- PJ42754 – ECB resource monitor enhancements
  - Ability to monitor groups of ECBs
  - Ability to profile resource usage

Future

Disclaimer

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

Tape redirect

Tape redirect

- Purpose: Improve the ability to consume data that is sent off of TPF using tape without changing legacy applications
- Provide ability to direct data that is written using TPF tape APIs to either:
  - MQ queue
  - File system file
- Only for output tapes
- Initially only for general tapes

Tape redirect

- Design thoughts
  - Data writes will be put on tape queue
  - A daemon will pull data from tape queue and write to specified location (one location only)
- Questions:
  - Have output location in tape label record?
    - Limited space in tape label mask records
  - Have output location on ZTMNT command?
    - Could be a lot of typing
    - Auto mount will not be able to specify output location

Tape redirect – Sponsor users

- Making design decisions now
- If you are interested in becoming a sponsor user, please contact your CSR

Summary

- Investing in several key areas
  - Performance
  - Scalability
  - Diagnostics
  - Operability

Thank you!

Questions or comments?

Trademarks