PJ46309 – Trace log enhancements

2022 TPF Users Group Conference
March 27-30, Dallas, TX
Systems Control Program

Michael Shershin

IBM Z

IBM

Agenda

- PJ46309 – Trace log enhancements
- PJ46309 - Additional trace enhancements

PJ46309 was released in November 2021

Problem Statement

When doing problem investigation, performance analysis, or enhancements, there is a need to understand the program path and resources that are used in z/TPF by a message. Existing tools can be used to get some information but key pieces of information are missing.

Pain Points

- When trace log is started, it only collects trace information for traces that are currently active.
- When trace log is started, it does not collect information for child ECBs.
- Trace log controls can unnecessarily limit the amount of data that is collected.

Technical Details - Infrastructure

A method is needed to use new options and to pass additional information when trace log is started and stopped

- TLOGC macro has a new parameter called PARMS.
  - Pointer to a structure that contains options and information to be passed.
  - Macro idstlg.mac describes the parameter area
- New C function tpf_tracelog_ext() is the C equivalent to TLOGC with the PARMS parameter.
  - C header tpf/idstlg.h describes the parameter area

Technical Details - Infrastructure

```
          BEGIN NAME=QZZ0,VERSION=00,DSECT=WORKAREA,USEREG=R4,BASE=R8
TLOGC_PARM DS  XL(ITLGLEN)
          APSTKC
          ...
          IDSTLG REG=R2
          LA    R2,TLOGC_PARM       Point to parameter block
          LR    R10,R2                "   "     "        "   for MVCL
          LHI   R11,ITLGLEN         Get the size of the parameter block
          LHI   R14,0               Set to zero
          LHI   R15,0               Set pad to zero
          MVCL  R10,R14             Zero the parameter block
          MVI   ITLGTYP,ITLGTYP_START_PROC TLOGC START,PROC
          LARL  R14,DIR             Point to directory name to use
          MVC   ITLGDIR(L'DIR+1),0(R14) Copy in directory name
          TLOGC START,PARMS=(R2)    Start trace log
          ...
          DS    0H
DIR       DC    C'/tmp'             TLOGC directory to use
          DC    X'00'               String end
```

Technical Details - Infrastructure

Example of tpf_tracelog_ext()

```
#include <tpf/tpfapi.h>

IDSTLG interface_buffer;
char  *mytext = "Format 2 global keypointing";

memset(&interface_buffer, 0, sizeof(IDSTLG));
interface_buffer.itlgsize = sizeof(IDSTLG);
interface_buffer.itlgtyp  = ITLGTYP_START_PROC;
interface_buffer.itlgto   = 0;
strcpy(interface_buffer.itlgdir, "/tmp");
strcpy(interface_buffer.itlgdestext, mytext);
tpf_tracelog_ext(ITLG_START, &interface_buffer);
```

Technical Details – Traces used

- Prior to PJ46309 ECBs in trace log sessions collect trace data based on the currently active set of ECB traces
- With PJ46309 ECBs in trace log sessions collect trace data based on a predefined set of ECB traces
  - ECBs that are in trace log sessions ignore the currently active set of ECB traces
  - ECBs that are not in a trace log session use the currently active set of ECB traces

Technical Details – Traces used

- The predefined set of traces are:
  - ECB macro trace
  - C function trace
  - Extended C function trace
  - Skip C function trace is not active for the ECBs in the trace log session
    - All C functions are traced for ECBs in a trace log session
  - Enter / back trace
  - z/TPFDF enter / back trace
  - Copy-on-write trace

Technical Details – Child ECB support

Trace log has the option to automatically start a trace log session for a child ECB.

- Option is specified when the trace log session is started.
- If the option is used, all child ECBs will automatically have trace log started. All grandchild ECBs will have trace log started. All great-grandchild ECBs will have trave log started. And so on.
- To enable trace log for child ECBs, set the ITLGOPT_CHILD indicator in the PARMS structure.

```
Assembler:
         OI    ITLGOPT,ITLGOPT_CHILD Start trace log for child ECBs
```

Technical Details – Child ECB support

- A child ECB is created when one of the following APIs are called.
  - CREMC or cremc()
  - CREDC or credc()
  - CREXC or crexc()
  - CREEC or creec()
  - CRESC or tpf_cresc()
  - SWISC TYPE=CREATE or swisc_create()
  - fork() and tpf_fork()
  - pthread_create()

Technical Details – Controls – Time out value

Trace log has the option to automatically stop collecting data after a specified number of seconds

- The option is a time out value that is specified when the trace log session is started
  - A value of zero indicates that there is no time out value
- The time out value is intended to limit the amount of data that is collected
- To use the time out value, set field ITLGTO in the PARMS structure to a non-zero value

```
Assembler:
         MVHHI ITLGTO,2            Set time out to 2 seconds
```

Technical Details – Controls – Time out value

The option to use a time out value is a method to limit the generations of child ECBs that collect trace log data

- The clock for the time out value is started when trace log is started for the first ECB
- The parent and all generations of child ECBs time out at the same time.
  - The time is determined by the start time of the first trace log session plus the time out value in seconds.

Technical Details – Controls – IUOWID management

With PJ46309 the IBM unit of work ID (IUOWID) is associated with the trace log session

- Association is done automatically
- If trace log is active for an ECB and if the IUOWID is changed, the trace log session for that ECB is stopped
  - Trace log is stopped when the next trace activity happens after the IUOWID is changed
  - If the IUOWID is changed, this is an indication that the ECB is taking on a different piece of work
  - Intended for long running ECBs that do disparate pieces of work

Technical Details – Controls – IUOWID management

Child ECBs inherit the IUOWID from the parent ECB

- If trace log is started for the child ECB and the IUOWID is changed for the child ECB, trace log is stopped for the child ECB
  - Trace log is stopped when the next trace activity happens after the IUOWID is changed

Technical Details – General controls

Before PJ46309 there were two sets of controls

- Maximum number of concurrently active trace log sessions
  - Controlled by ZASER TRLOG
  - Typically set to 0, which means that no trace log sessions are allowed
- Maximum amount of system heap that is allowed to be on a trace log queue
  - Controlled by ZASER TRLOGTH
  - The default is 50 MB

Technical Details – General controls

With PJ46309 controls are based on the availability of 1 MB frames (FRM1MB)

- The previous controls have been obsoleted
- With PJ46309 1 MB frames are the primary resources used by trace log
- At trace log start, a LODIC check is done using CLASS=IBMBATCH to determine if ECBs and 1 MB frames are acceptable. If not, the trace log session is not started
- When trace log data is queued, a LODIC check is done using CLASS=IBMBATCH to determine if 1 MB frames are acceptable. If not, the trace log session is stopped.
  - LODIC CHECK,CLASS=IBMBATCH,CHECKONLY=(FRM1MB)

Technical Details – General controls

What does 1 MB frames have to do with trace log?

- Trace log copies ECB trace entries into a buffer and when the buffer is full it is put on a queue to be processed later.
- The buffer is obtained from 64-bit system heap in 1 MB units.
  - If the 64-bit system heap is not obtained from the preallocated area, 1 MB frames are used.
  - When system heap that was obtained in 1 MB units and not from the preallocated area is returned, the 1 MB frames are returned to the system.
  - System heap has been updated so that trace log buffer requests do not use the preallocated area; 1 MB frames are always used

Technical Details – General controls

What happens when a trace log session is stopped by a control?

- Stop collecting data
- The data that has been collected and is on queue is processed
- The output will contain partial data
  - All trace data that was collected up to the point when trace log was stopped is processed and included in the output
  - Any trace data after the controls stopped trace log is not part of the output

Technical Details – Compression

On a z15 processor compression is used

- When trace log copies trace data into 64-bit system heap in 1 MB units
  - Over 3,600 trace entries can fit into 1 MB
- On a z15 when the 1 MB buffer is full it is compressed into another system heap buffer that is 1 MB in size.
  - IBM TPF lab test results were the 1 MB buffer compresses to about 150 KB. Your results might vary.
- When the compressed 1 MB buffer is full, it is added to a queue to be processed later
- Compression is effective for ECBs with a large number of trace entries

Technical Details – Resource usage counts

With PJ46309 Resource usage counts are collected by trace log

- Information for 65 different metrics are collected
  - Same information collected for runtime metrics collection database
- Counts are collected when the trace log session starts
- Counts are collected when an ECB owner name change happens
- Counts are collected when the trace log session ends
- The .report output file shows:
  - Counts used during the trace log session
  - Counts at the start and end of the trace log session

Agenda

PJ46309 – Trace log enhancements

PJ46309 - Additional trace enhancements

PJ46309 was released in November 2021

Pain Points

- ECB trace does not report when a copy-on-write happens.
- Important information is not collected in macro trace.

Technical Details

All additional trace information is available in the following:

- Dumps
- Trace log
- Output from command ZDECB TR

Technical Details

Copy-on-write trace is available

- When a copy-on-write happens, an entry is put into macro trace.
- The trace entry includes:
  - The EVM address of the 4K page that was copied
  - The location of the instruction that caused the copy-on-write
- Here is an example.

```
IBM_DEFT    QZZ5YZ LOADSET-DRIVER01 OBJECT-qzz5
 64PU1  1       A0 QZZ5 C-O-W          PG-000000009EB19000   DAA8C230 1320E86A
                                A-000000009EB16A58
```

Technical Details

Copy-on-write trace is controlled using the ZSTRC command

- Here is an example to turn on Copy-on-write trace:

```
ZSTRC ALTER COPYWRT
CSMP0097I 17.03.19 CPU-B SS-BSS  SSU-HPN  IS-01
STRC0013I 17.03.19 SYSTEM TRACE OPTIONS
TYPE OF SYSTEM TRACE          KEYWORD  STATUS
 ECB TRACE                    TRACE      ON
 FUNCTION TRACE               FUNCTR     ON
 EXTENDED FUNCTION            FUNCEXT    ON
 SKIP FUNCTION                FUNCSKIP   OFF
 TPF ENTER/BACK TRACE         ENTER      ON
 TPFDF ENTER/BACK TRACE       DFENTER    ON
 IO TRACE                     IO         ON
...
 TEST AUTOMATION              TESTAUTO   ON
 COPY-ON-WRITE TRACE          COPYWRT    ON
END OF DISPLAY+
```

Technical Details

ECB time slice is in macro trace

- When an ECB gives up control because of a time slice, an entry is put into macro trace.
- Collection is done when ECB macro trace is active.
- Here is an example.

```
IBM_DEFT    QZZ5YZ LOADSET-DRIVER01 OBJECT-qzz5
 64PU1  1       B0 QZZ5 SUSPND  ECB SUSPENDED BY TIME SLICE  DAA8C230 135B00A4
 64PU1  1       AC QZZ5 SUSPND  ECB SUSPENDED BY TIME SLICE  DAA8C230 13D88898
```

Technical Details

Macro parameters have been enriched for various macros

- CEBIC: the subsystem name and subsystem user are added for HPO systems.
- CINFC: the request type (R, W, or K) and the name are included.
- CORHC: an indication of whether the resource was held by another ECB and the amount of wait time in microseconds are included.
- ENQC: an indication of whether the resource was held by another ECB and the amount of wait time in microseconds are included.
- EOWNRC: the owner name that was set is included.

Technical Details

Macro parameters have been enriched for various macros (continued)

- File type macros: an indicator is included that reports whether the record was only put into VFA (e.g. delay file) or whether the record was written to DASD.
- Find type macros: an indicator is included that reports whether the record was retrieved from VFA or DASD.
- SANQC: an indication of whether the resource was held by another ECB and the amount of wait time in microseconds are included.

Technical Details

Example

```
IBM_DEFT    QZZ0YZ LOADSET-DRIVER01 OBJECT-qzz0_CORHC_FIWH
 31PU1  1      29A QZZ0 CINFC  R NAME=CMMIST                 DB0A83EB 36B39240
 31PU1  1      2A2 QZZ0 EOWNRC                              DB0A83EB 36B399AE
 31PU1  1      2A4 QZZ0  OWNER  CHANGE ITEST   QZZ5    CHILD DB0A83EB 36B3A2A0
                          ECB
 31PU1  1      2AE QZZ0 CEBIC  SS-BSS  SSU-HPN               DB0A83EB 36B3B48C
 31PU1  1      2C2 QZZ0 EVNTC  CORHC N-000000000FC2E018      DB0A83EB 36B3BA34
                          HELD-Y T-2027186
 31PU1  1      2CE QZZ0 POSTC  CORUC N-000000000FC2E018      DB0A83ED 259F15B6
 31PU1  1      2F6 QZZ0 FIWHC  F-00000000D4030000 L-D2       DB0A83ED 259F6B3A
                          ID-C1C1 VFA
 31PU1  1      2FE QZZ0 FILUC  F-00000000D4030000 L-D2       DB0A83ED 2710016A
                          ID-C1C1 DASD
```

Value Statement

When debugging a problem, ECB trace provides greater insight into the system services that the application used.

When trace log is run, more complete information can be obtained. As a result a more thorough understanding of the path and resource usage for a z/TPF message can be gained.

Thank you