Control program updates and enhancements

Michael Shershin
TPF Development lab

Agenda

- PJ44596 – FARF6 fixed file support (PUT 14)
- PJ44680 – Active program detection
- PJ44633 – New format 2 global open option
- PJ44697 – ZWIPL – Write IPL records
- PJ44837 – Change pool ratio dispensing without an outage
- PJ44907 – ECB heap changes
- PJ45014 – Better performing C function for format 1 globals
- What’s next

PJ44596 – FARF6 fixed support

- Supports the allocation and use of FARF6 fixed record types
  - All record sizes can be used
    - Small, large, 4 KB
  - All types of uniqueness can be used
    - SSU unique, processor unique, I-stream unique
  - Highest ordinal number supported is 4,294,967,295
  - Must use DECBs
  - Must use FAC8C or FACZC to get the fixed file addresses

- Can use FARF6 fixed records for a fixed file system (FFS)
  - FFS is processor unique

Agenda

Problem

The program base contains thousands of programs. Many programs are 20, 30, even 40 years old. It is not known which programs are still in use.

When projects are done, programs that are no longer used may be updated. Resources are wasted updating, reviewing, testing, and implementing unused programs.

With PJ44680

TPF has ability to identify which programs have been entered at least once. These programs are in use.

The set of programs that have not been entered needs additional investigation to determine whether the programs maybe used.

- There may be programs that are only entered in special circumstances.
  - Pool reallocation or database reorganization programs are examples.
- Another scenario is that programs may only be entered when errors happen.

PJ44680 – Active program detection

- Always collecting status
- Indicators in PAT entry for each program provide status
  - When a program is fetched, PATCFG_APD_FETCHED indicator is set
  - When a program is entered, PATCFG_APD_CALLED indicator is set
  - Program with special linkage (i.e. CTAL) do not have status collected
    - Assume that these are always used
- Once an hour, PAT indicators are written to file system status file
  - Processor unique, binary file
  - For CPU-A file is: /etc/tpf-apd/.status_A
- New command ZAPDC creates a report
  - Creates a comma separated variable (csv) file

    ➔ZAPDC REPORT FILE-*file_name*

PJ44680 – Active program detection report example

```
ACTIVE PROGRAM DETECTION REPORT, OUTPUT_VERSION=1
Creation Date Fri Mar 16 22:56:27 2018

Active Program Detection Start Dates
CPUID, Date
B,     Mon Oct 23 13:52:48 2017
C,     Mon Oct 23 13:58:04 2017
D,     Mon Oct 23 13:58:04 2017

Active Program Detection Status
Program Name, Type, Any,     B,       C,       D
CISO,         CSO,  OTHER,   OTHER,   OTHER,   OTHER
UCST,         BSO,  FETCHED, FETCHED, FETCHED, FETCHED
CTAL,         CSO,  OTHER,   OTHER,   OTHER,   OTHER
CLBM,         CSO,  CALLED,  CALLED,  CALLED,  CALLED
CFVS,         CSOE, CALLED,  CALLED,  CALLED,  CALLED
BKB0,         BSO,  CALLED,  CALLED,  NO,      NO
BRU3,         BSO,  NO,      NO,      NO,      NO
CVOE,         BSO,  CALLED,  NO,      CALLED,  CALLED
```

PJ44680 – Active program detection additional information

- Status is saved across an IPL
  - In restart PAT indicators are updated from status file
- Ability exists to reset status
  - An IPL is required for the reset to take effect
    - IPL sets linkage so that PAT indicators are updated on fetch and enter
  - Command: → ZAPDC RESET

PJ44680 – Automatically set ECB owner name on enter to program

- With PJ44680 support for program configuration files exist
  - IBM program configuration file:
    - On linux: base/cntl/ibmconfig.csv
    - On TPF: /sys/tpf_pbfiles/apps/pcfg/ibmconfig.csv
  - User program configuration file:
    - On linux: <sys>/cntl/usrconfig.csv
    - On TPF: /sys/tpf_pbfiles/apps/pcfg/usrconfig.csv
  - Comma separated variable (csv) files

- On enter to programs in the program configuration file, an EOWNRC is called to set the owner name that is specified in the file.
- Eliminates the need to update the program to call EOWNRC

PJ44680 – Automatically set ECB owner name on enter to program

- Owner name is fixed; cannot dynamically decide what owner name to use
  - Use the owner name that is in the program configuration file
- Contents of program configuration file are used:
  - When loadset containing the file is activated
  - If TLDR loaded, when the processor is IPLed on the image where the file was loaded
- Example, IBM program configuration file contains

```
1
CVFE,IGLOBALS.KEYPOINT.FMT1
CDTF,ICLOCKS..
CEL5,IOLDR.POLICE.
```

- On enter to program CVFE, owner name IGLOBALS.KEYPOINT.FMT1 is set

PJ44680 – Automatically set ECB owner name on enter to program

- PJ44470, automatic owner name restore, was released in early 2017
- Combine PJ44680 and PJ44470
  - Automatically set an owner name on enter to a package
  - Automatically restore owner name to previous value when leaving the package

- Eases the ability to identify resource usage by not requiring program updates to set the ECB owner name
  - Resource usage by owner name (ZMOWN)
  - Name-value pair collection

Agenda

PJ44596 – FARF6 fixed file support

PJ44680 – Active program detection

PJ44633 – New format 2 global open option (PUT 14)

PJ44697 – ZWIPL – Write IPL records

PJ44837 – Change pool ratio dispensing without an outage

PJ44907 – ECB heap changes

PJ45014 – Better performing C function for format 1 globals

What’s next

PJ44633 – New format 2 global open option

- Options to open a format 2 global are:
  - TPF_GLRD - open for read only; a global descriptor is returned
  - TPF_GLRDWR – opened for read and write; a global descriptor is returned
    - Obtains lock on format 2 global record
  - TPF_GLRDFST – opened for read only; no global descriptor is returned
    - Cannot change from read only to read and write
  - <u>New option</u>: TPF_GLRWTNLK – opened for read and write; a global descriptor is returned
    - A lock is not obtained
    - Users must do their own serialization if updates are made to the global
    - Only allowed for keypointable format 2 globals
      - » Globals defined as KEYPOINT=YES,PROC=YES,SYNC=NO

PJ44633 – Value of TPF_GLRWTNLK

- Using a format 2 global that is keypointable and I-stream shared
  - ➔ ZGLBL GLOBAL DEFINE MYF2GLB LOC-64 KEY-YES PROC-YES SYNC-NO IS-NO
  - ➔ ZGLGL GLOBAL INITIALIZE MYF2GLB SOURCE-ZERO SIZE-1000
- Assume format 2 global holds counts
  - When open option TPF_GLRDWR is used, a CORHC is used to serialize updates for keypointable global
  - Assume counts are updated hundreds / thousands of times a second
    - When multiple I-streams are in-use, contention happens on the CORHC
    - Contention causes large numbers of ECBs to wait on CORHC
- Must file (keypoint) the global in order to save counts across an outage
  - Before PJ44633, only option is to open using TPF_GLRDWR and close with update (tpf_glClose option TPF_GLUPD)

PJ44633 – Value of TPF_GLRWTNLK

– Before PJ44633, code to update the counts may be:

```
struct my_global {
    unsigned long count;
 };

 int               glDescr;
 struct my_global *pMYGLB;

 glDescr = tpf_glOpen("MYF2GLB ", TPF_GLRDWR, (void *) &pMYGLB);
 pMYGLB->count++;
 tpf_glClose (glDescr, TPF_GLUPDWT);
```

PJ44633 – Value of TPF_GLRWTNLK

- With PJ44633, open option TPF_GLRWTNLK may be used
  - The CORHC is not issued; the global is not locked
  - CORHC contention does not happen
  - ECBs are not tied up waiting to hold the CORHC lock
  - But, the user must do their own serialization when updating fields in the global

PJ44633 – Value of TPF_GLRWTNLK

– With PJ44633, code to update the counts may be:

```
struct my_global {
    unsigned long count;
 };

 int               glDescr;
 struct my_global *pMYGLB;

 long              newValue;
 long              oldValue;

 glDescr = tpf_glOpen("MYF2GLB ", TPF_GLRWTNLK, (void *) &pMYGLB);
 do {
   oldValue = pMYGLB->count;
   newValue = oldValue+1;
 } while (csg(&pMYGLB->count, &pMYGLB->count, newValue) != 0);
 tpf_glClose (glDescr, TPF_GLUPDWT);
```

Agenda

- PJ44596 – FARF6 fixed file support
- PJ44680 – Active program detection
- PJ44633 – New format 2 global open option
- PJ44697 – ZWIPL – Write IPL records (PUT 14)
- PJ44837 – Change pool ratio dispensing without an outage
- PJ44907 – ECB heap changes
- PJ45014 – Better performing C function for format 1 globals
- What’s next

PJ44697– ZWIPL – Write IPL records

- PJ42031 (PUT 11) removed the requirement for ESA/390 architecture
  - IPL2 was updated to support use of z/Architecture on CPU resets and IPL
  - To load IPL2, a loader general file IPL was required
- z14 machines do not support ESA/390 architecture
  - In order to IPL z/TPF on a z14, PJ42031 version of IPL2 must be used

- PJ44697 provides an alternative method to load IPL2 changes
  - New command ZWIPL is provided
  - ZWIPL writes IPL1 and IPL2 to the specified SDA
  - ZWIPL can only run on processor that owns WIPL in the processor ownership table (PROT)

➔ ZPROT ADD UT WIPL BSS

➔ ZPROT ASN UT WIPL BSS

Agenda

PJ44596 – FARF6 fixed file support

PJ44680 – Active program detection

PJ44633 – New format 2 global open option

PJ44697 – ZWIPL – Write IPL records

PJ44837 – Change pool ratio dispensing without an outage (PUT 14)

PJ44907 – ECB heap changes

PJ45014 – Better performing C function for format 1 globals

What’s next

PJ44837 – Change pool ratio dispensing without an outage

- Before PJ44837
  - The following commands updated file records only. To take effect, an outage is required.
    - Update pool ratio dispensing: → ZGFSP RTO
    - Update pool fallback schedule: → ZGFSP FLB
- With PJ44837
  - Ratio dispensing and fall back schedule updates take effect immediately
    - On all processors in the loosely coupled complex
    - No changes to the commands

Agenda

- PJ44596 – FARF6 fixed file support
- PJ44680 – Active program detection
- PJ44633 – New format 2 global open option
- PJ44697 – ZWIPL – Write IPL records
- PJ44837 – Change pool ratio dispensing without an outage
- PJ44907 – ECB heap changes (PUT 14)
- PJ45014 – Better performing C function for format 1 globals
- What’s next

PJ44907 – ECB heap changes

- Observing increase in ECB heap usage
  - IBM and user programs
  - Java is one example
- Reduce overhead for ECBs and processes that get a large number of unique ECB heap buffers
  - ECB control table overflow handling is updated
  - Changes happen automatically
    - No tuning required
    - No application changes required

PJ44907 – ECB heap changes

- Changes to handling of overflow entries in the ECB heap control table
  - Before PJ44907
    - Preallocated ECB heap control table has 151 entries
    - If more than 151 entries are needed, a 4 KB system heap buffer is obtained
      - Allocates an additional 31 entries
      - Additional 4 KB system heap buffers are obtained as needed
    - If large numbers of entries in the control table are needed (i.e. large number of unique ECB heap buffers), overhead exists in managing system heap for overflow handling, especially at EXITC time.

PJ44907 – ECB heap changes

- Changes to handling of overflow entries in the ECB heap control table
  - With PJ44907
    - Preallocated ECB heap control table has 151 entries (no change)
    - If more than 151 entries are needed, a 32 KB system heap buffer is obtained
      - Allocates an additional 255 entries
      - Additional 32 KB system heap buffers are obtained as needed
    - Reduces overhead in managing system heap

Agenda

- PJ44596 – FARF6 fixed file support
- PJ44680 – Active program detection
- PJ44633 – New format 2 global open option
- PJ44697 – ZWIPL – Write IPL records
- PJ44837 – Change pool ratio dispensing without an outage
- PJ44907 – ECB heap changes
- PJ45014 – Better performing C function for format 1 globals (PUT 14)
- What's next

PJ450214 – Better performing C function for format 1 globals

- New C function: glob_fast()
  - Generates in line code to return the address of a format 1 global

- Alternative to C function: glob()
  - Service routine is in the CTAL library
  - Incurs linkage overhead to return the address of a format 1 global

- Example:

```
#include <tpf/tpfglbl.h>
#include <tpf/c_globz.h>

int *lastMidnight_ptr;

lastMidnight_ptr = (int *)glob(_u1mid);
lastMidnight_ptr = (int *)glob_fast(_u1mid);
```

Agenda

Disclaimer

What’s next

- Increase capacity for DASD I/Os per second (IOPS) on an individual LPAR
  - Goal is to at least double current capacity

- Constraint relief for memory below 2 gig
  - Creation of a new core block type above 4 gig

- Improve performance when activated loadsets are in use
  - Focus on activated programs that are entered frequently

- Provide ability to protect pool addresses that were made available by a PDU
  - Minimize damage in the case when lost addresses are returned by mistake

Thank you

Michael Shershin

shershin@us.ibm.com