z/TPF V1.1

TPF Users Group - Spring 2009
z/TPF Migration Experiences

Name: Kevin Jones
Venue: Main Tent

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Agenda

- **Purpose and Background**
- **Lessons Learned**
  - System Generation and Build
  - Application Programs
  - Local Modifications
  - Application and System Testing

Agenda

- Tooling
- System Settings and Tuning
- Source Code Manager (SCM)
- Education
- Project Planning
- **Key Messages and Conclusion**
- **Question and Answer**

Purpose and Background

- **The purpose of this presentation is to share IBM's experiences with migrating TPF 4.1 systems to z/TPF**
- **IBM has been involved with many completed and in-progress migrations**
  - Gives us a great deal of practical experience to help better support all of our customers

Purpose and Background

- **IBM has also conducted the complete migration of a production TPF 4.1 system to z/TPF:**
  - Planning
  - Establishing the development environment
  - Updating applications and local modifications
  - Application and system testing
  - Education
  - Planning and executing the cutover
    - fallback scenarios also, which were not needed

Purpose and Background

- **We had a unique opportunity to “live” a migration from it’s earliest planning stages through cutover**
  - 35 changes were made to our various products as a result of this experience
- **The “lessons learned” presented here are drawn from these experiences**
- **This presentation is intended to supplement other TPFUG “migration experience” presentations from our customers**

Lessons Learned – System Generation and Build

- **The "z/TPF Migration Guide" is a key document and should be followed closely, especially in terms of updating you SIP stage 1 deck**
  - See also the Fall 2008 TPF Users Group presentation entitled "z/TPF Systems Generation"
- **It is not necessary to build the GCC compiler**
  - binaries provided by Red Hat work correctly

Lessons Learned – System Generation and Build

- **Defining a precise, well-documented "promotion" process for integrating new and changed code is critical**
- **Ensure that proper networking access is available, including firewall authorizations, between your various systems:**
  - workstations, z/VM, z/OS, Linux, SCM, z/TPF
  - In-house networking changes may be needed!

Lessons Learned – System Generation and Build

- **Removal of versions from object code file names can be effectively replaced using:**
  - A source code manager such as Clearcase
  - Object code comments specified in the maketpf “config” file (variable TPFOBJPP_COMMENT)
    - Can be displayed online using ZDMAP, and offline using the “tpfzdmap” utility
    - Details can be found in the Fall 2007 presentation “Code Flow and Versioning”

Lessons Learned – System Generation and Build

- **Keeping the HFS structure simple is important**
  - A small modification to the z/TPF directory structure had several unforeseen impacts
- **Having Group IDs (GIDs) and Users IDs (UIDs) match between Linux and z/OS will simplify your environment**
- **Building application or system code while running as “root” under Liunx or z/OS Unix System Services (USS) is not advised**

Lessons Learned – System Generation and Build

- **The “loadtpf” command on Linux is extraordinarily useful**
  - Provides the ability to create OLDR or TLDR output to a file
  - The file can then be FTP’d to z/TPF and loaded
  - Eliminates the need to use physical media such as tapes, although the ability to write to tapes is available from Linux
  - Greatly reduces the need to run loaders from z/OS (although ALDR must still be run there)

Lessons Learned – System Generation and Build

- **The date for freezing the z/TPF product base should be driven by need, not PUT deliveries**
  - It is not necessary to freeze on a PUT boundary, especially given their yearly delivery
  - All z/TPF and z/TPFDF APARs are assigned a unique sequence number upon closure, allowing maintenance to be applied to any given point
    - The following site contains useful APAR data:
      - http://ibm.com/tpf/maint/maintztpf.html
  - IBM chose to freeze maintenance at the beginning of application and system testing, when PUT 5 was not yet closed
    - Effectively created a PUT level for the customer

Lessons Learned – Application Programs

- **Assembler programs are generally easy to convert to z/TPF using the TPF Toolkit**
  - 600 programs scanned and updated in two days (1.6 minutes per program)
  - Updated programs assembled immediately to detect new warnings from stricter assembler
  - No applications were written in C/C++
- **The TPF Toolkit single-source scans should be used even if a single application program base will not be maintained**
  - In all cases, a process is required to ensure that non-compliant code is not introduced after the initial scans

Lessons Learned – Application Programs

- **24-bit applications are problematic**
  - Problems are not easy to detect
  - Even 31-bit applications can have 24-bit errors
    - In TPF 4.1, core blocks and primary globals were mapped under the 16MB line. In z/TPF, they are only mapped under 2GB.
  - Best approach is to start converting 24-bit applications early, and possibly make the same changes on TPF 4.1 to get additional testing

Lessons Learned – Application Programs

- Once a 24-bit defect is found and corrected, search all of your applications for similar programming (register or field usage)
  - This will likely yield additional changes
- Also examine displays generated from your application programs for 24-bit fields (six-digit core addresses) which need to be expanded
  - Scanning for “TR” instructions with a length of 6 bytes, and “UNPK” instructions with a length of 7 bytes may be helpful

Lessons Learned – Local Modifications

- **Every local modification to the TPF product must be examined**
  - First, determine if the mod is necessary in z/TPF, given the new functionality available
  - If the modification is needed, determine if it can be moved to a user exit
    - z/TPF has nearly 300 user exits!
  - CP modifications require special attention since it now runs in 64-bit mode
    - Pay particular attention to addresses which may be shared between base product code and the local modification

Lessons Learned – Application and System Testing

- **The importance of testing during a z/TPF migration cannot be over-emphasized**
  - It is not unusual for testing to consume 50% or more of all hours spent

Lessons Learned – Application and System Testing

- **Significant testing is required even if minimal changes are made to applications and local modifications**
  - The underlying operating system is completely revamped
  - Application behavior and timing may be affected
  - Expect to find base application errors because of the new environment and such extensive testing
    - Latent defects were found in 2.3% of all “working” applications

Lessons Learned – Application and System Testing

- **Types of testing to consider:**
  - Unit test of all local modifications
  - Repetitive regression tests of all applications
  - Regression test of z/TPF functionality to be used
  - Overall system testing (concurrent workloads)
  - Long-term stability tests
  - Stress tests to measure performance and tune
  - Disaster recovery sites
  - Hardware failure scenarios
  - Backup hardware (processors, tape drives, etc)
  - End user testing

Lessons Learned – Tooling

- **Use of tooling during a z/TPF migration is critical:**
  - TPF Toolkit
    - Single source scans, including the ability for customers to define their own rules
    - New dump viewer allows interactive debugging
    - Integrated TPF Debugger
  - Continuous Data Collection
    - realtime system status and resource usage

Lessons Learned – Tooling

- Software Profiler
  - used to understand how applications and the system are affecting overall performance
  - detect bottlenecks and fine tune performance
- Automation Platform such as TPF Operations Server (TOS)
  - provides the ability to automate repetitive tasks and test variations

Lessons Learned – System Settings and Tuning

- **Core allocations (SIP stage 1 macro CORREQ, and command ZCTKA)**
  - Number of IOBs, SWBs, common blocks, 4K frames and ECBs should be set initially to their 4.1 values
  - Preallocated ECB heap should be initially set based on the TPF 4.1 Data Reduction report “ECB Heap Area Usage Summary”
    - set “PEH” so that 90% of ECBs will have heap requests satisfied from the preallocated area

Lessons Learned – System Settings and Tuning

- Set initial 1MB frame allocation to:
  - ((10% number of ECBs) + (number of 1MB frames needed for core resident programs)) * 2
    - This is likely an over-allocation that can be corrected as testing proceeds
- Minimum VFA and 31-bit system heap (SHP) sizes should be initially set based on 4.1 usage
- The default value for the dump buffer area (DBA) is an appropriate initial setting
- As testing proceeds, storage allocation values should be adjusted based on actual usage

Lessons Learned – System Settings and Tuning

- **System Trace Options (ZSTRC)**
  - Options that have been carried over from TPF 4.1 can retain their settings
  - New options for z/TPF need to be carefully examined
  - Note that several ZSTRC options are excellent debugging tools and should be enabled for portions of application and system testing
    - Block check, heap check and branch relative target check

Lessons Learned – System Settings and Tuning

- **Dump Options (ZASER)**
  - Options that have been carried over from TPF 4.1 can retain their settings
  - New options for z/TPF need to be carefully examined, although the default values are acceptable as initial settings
    - Particularly true for new parameters which help control the size of dumps
    - Depending on your memory size and dump frequency, you may or may not need to limit dump sizes

Lessons Learned – Source Code Manager (SCM)

- **Rational Clearcase is being used as an SCM for z/TPF systems, among others**
  - All features of Clearcase can be exploited, particularly from the command line
    - Use of the remote client does require special considerations
    - Clearcase can also be used in simpler forms tailored to need
  - The TPF Toolkit can be customized to interact with Clearcase

Lessons Learned – Education

- **DO NOT diminish the importance of education in your project planning!**
  - z/TPF functionality
    - Operational changes
    - New and changed APIs
    - New and changed utilities
    - Debugging, including new traces and dump formatting
  - 64-bit programming

Lessons Learned – Education

- Tooling options
- Linux skills
- z/OS Unix System Services (USS)
- SCM skills
- Maketpf, bldtpf and loadtpf
  - Config, control and make files
- **Not every person needs every skill, but proper up-front education should pay for itself**

Lessons Learned – Education

- **There are many sources of information:**
  - User Group presentations back through 2005
    - IBM and non-IBM
  - z/TPF Migration Guide
  - z/TPF Information Center
  - Internet search engines
  - Formal education offerings
  - TPFUG Migration Task Force conference calls
- **Education also needs to be on-going, as new function is being added regularly to z/TPF**

Lessons Learned – Project Planning

- **12 to 24 month migration plans are typical**
- **IBM spent 7.5 person-years to migrate a production TPF 4.1 system to z/TPF**
  - However, *many* factors affect the amount of resource needed, both negatively and positively
  - Further, some costs are fixed regardless of system size, such as installing Linux

Lessons Learned – Project Planning

- **The following is how the resources were used, which may or may not apply to other migrations:**
  - Application and system testing – 50%
  - Forward-fit and test of local modifications – 15%
  - Project management – 10%
  - Establishing a development environment – 10%
  - Development of test tools and automation – 5%
  - System generation and maintenance – 5%
  - Host systems (Linux, VM, z/OS) – 3%
  - Application migration and support – 2%
    - Atypically low given the lack of C/C++ applications

Lessons Learned – Project Planning

- **Some additional points to consider:**
  - Remember that test applications and automation scripts need to be updated also
  - Include time to update in-house procedures, processes and user guides
  - Ensure your new development environment is replicated as needed at your disaster recovery sites
  - Spend time to ensure your test plans are complete, and cover all of your applications
    - Watch for undocumented features
    - Underscores the need for end-user testing

Key Messages and Conclusion

- **z/TPF migrations are no longer a theoretical exercise – they are being *completed* successfully!**
- **Simplicity in your development environment is important**
  - Weigh carefully the cost versus benefit of adding complexity
- **The z/TPF Migration Guide and other information sources are vital, and represent proven migration approaches**

Key Messages and Conclusion

- **Tooling is critical to success**
  - automation, debugging, system and application analysis, etc.
- **Work with IBM on any requirements you may have against our base products**
- **Education is vital and should be included in migration project plans**
- **IBM is ready and able to work with our customers to ensure continued success!**

Question and Answer