z/TPF EE V1.1
z/TPFDF V1.1
TPF Toolkit for WebSphere® Studio V3
TPF Operations Server V1.2

IBM Software Group

TPF Users Group Fall 2005

z/TPF File Systems - New & Improved in z/TPF !

Name : Stephen Record
Venue : Database / TPFDF Subcommittee

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

Trademarks

IBM, WebSphere, z/TPF, and z/TPFDF are trademarks of International Business Machines Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Notes
Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.

Agenda

- z/TPF file systems
- File service levels
- File attributes
- New and changed commands
- New C language APIs
- Summary comparison

z/TPF File Systems

- Four file systems in z/TPF
  - TFS - z/TPF Collection Support File System
  - MFS - Memory File System
  - FFS - Fixed-File File System
  - PFS - Pool File System

Hierarchical File System Fundamentals

- A file system consists of directories and files.
- A directory is a special file which contains pointers to other directories or files.
- Each directory and file in the file system tree is represented by a control block called an inode.
  - The inode resides on the dasd surface.
  - When the file is accessed, the inode is brought into memory to be the single point of control for the file.
- Each instance of a file system has a special directory which called the root directory "/".
  - The root directory is the base of the file system's tree of directories and files.
  - The TFS file system is always the root file system.

Virtual File System Overview

- The VFS is a code layer that provides the common API into the TPF file system support.
- It provides support that is common to all underlying file systems.
- The underlying file systems (TFS, MFS, FFS, PFS) interface with the VFS layer to provide the complete implementations of those file systems.
- Each underlying file system is implemented differently and has different characteristics.
- One file system may be better suited for a particular purpose than another.
- VFS support is processor and subsystem unique.

TFS - z/TPF Collection Support File System

- The renamed original file system in TPF4.1
- Still uses collection support and commit
- Processor shared and sub-system unique
- Interoperates with existing TPF4.1 file systems
- Supports inode and directory caching, and user named attributes for files
- Locking support is limited to exclusive full file locks
- Only one instance of the TFS can exist
  - It is the root file system
  - Mounting and unmounting is not supported

MFS - Memory File System

- New for z/TPF
- Uses system heap as file storage
  - Backed by 1 MB frames
- Does not persist over an IPL
- Processor and sub-system unique
- Supports record buffering, full file and byte range locking, and user named attributes for files
- Can be mounted and unmounted
- Typically mounted on /tmp and used by the system for temporary files.
- Instances of MFS are created on mount and deleted on unmount

FFS - Fixed-File File System

- New for z/TPF
- Processor and sub-system unique
- Both inodes and data blocks are allocated from a fixed-file record type (limited to 1 million fixed file records)
- Persists over IPL and unmount/mount processing
- Supports record buffering, inode and directory caching, full file and byte range locking, and user named attributes for files
- Uses core locking to serialize access to the file and its data
- Uses normal find/file
- Does not use commit

PFS - Pool File System

- New for z/TPF
- Processor and sub-system unique
- Inodes are allocated from a fixed-file record type but data blocks are allocated from pool records
  - Limited to 1 million fixed file records, ca. 15 million files
- Persists over IPL and unmount/mount processing
- Supports record buffering, inode and directory caching, full file and byte range locking, and user named attributes for files
- Uses core locking to serialize access to the file and its data
- Uses normal find/file
- Does not use commit

File Service Levels

- Record buffering is provided for use by FFS and PFS.
- It supports the reading/writing of file system records from/to a record buffer area shared by all processes and file systems on the processor.
- The use of the record buffering is controlled at the file level through an attribute called a File Service Level.
- Each File Service Level defines a set of parameters
  - How much of the record buffer, if any, may be occupied by records from the file
  - Whether writes are synchronous or not (writes are always synchronous in 1052 state)
  - How many changed records from the file may be buffered, if any, and for how long.

System File Attributes

- Support the setting and retrieving of named system file attribute values
  - The system attributes supported vary by file system type
    - File service level (FFS and PFS only)
    - Record IDs to assign
      - Data records (FFS, PFS, and TFS)
      - Object control records (TFS only)
      - Index records (TFS only)
      - Directory records (TFS only)
    - TPFCS DDNAME to use (TFS only)
- These attributes may be set at create time using the new tpf_open functions and may subsequently be changed or interrogated using the new file attribute functions or the ZFILE ATTR command.

User File Attributes

- All file systems support the capability for a user with correct permissions to assign user attributes of the form name=value to a file using the new TPF file attribute APIs.
- User File attributes can be interrogated, changed or deleted using the new ZFILE ATTR command or the new file attribute functions.

File System Check Utility

- Scandisk-like function with fix capability for all file systems (TFS, MFS, FFS, PFS)
- Invoked via the ZFILE FSCK comand
- Ability to check and optionally correct a file system while in use, without requiring a re-initialization of the file system or an IPL
- The actual checks and/or fixes performed depend on the implementation of fsck for that file system.
- Typical checks performed:
  - Scans for lost inodes (files or directories)
  - Scan for dangling directory entries
  - Scan for bad inode data

New and Changed Commands

- ZAVFS
  - New BUILD option to create or reinitialize FFS or PFS file systems
- ZAVFS and ZDVFS
  - New SERVICE option to modify or display File Service Table information
- ZDSMG DEFINE
  - Support the definition of a DDNAME which refers to a file system file
- ZFILE ATTR
  - Displays, sets, or removes the attributes of a file
- ZFILE FSCK
  - Performs filesystem checks and fixes on the specified file system

More New Commands

- ZFILE MOUNT
  - Supports the mounting of a file system
  - Records the mounting of the file system in a TPF record called the MTAB
  - File system will be automatically remounted after an IPL
  - Also used to change the mount options of a previously mounted file system (for example, from read-only to read-write or vice versa)
  - Mounting an MFS first creates a new MFS instance
- ZFILE UMOUNT
  - Supports the dismounting of a file system
  - Removes it from the MTAB
  - Dismounting an MFS also destroys the MFS instance

New C Language APIs

- File open functions
  - tpf_open()
    - Allow an application program to specify a file attribute structure at open time
    - Supported by all file systems.
  - tpf_openZdsmgDD() and tpf_fopenZdsmgDD()
    - Allow an application to specify both a file attribute structure and a ZDSMG DDNAME as the file path and the interface to issue the correct open to enable access to the data.
- File attribute manipulation functions
  - tpf_setFileAttribute() and tpf_fsetFileAttribute()
  - tpf_getFileAttribute() and tpf_fgetFileAttribute()
  - tpf_delFileAttribute() and tpf_fdelFileAttribute()

z/TPF File System Comparison

	FFS	MFS	PFS	TFS
Data store	Fixed file records	System heap	Pool records	z/TPFCS (pools)
Data persistence	Yes	No	Yes	Yes
Processor shared	R/O	No	R/O	Yes
Mountable	Yes	Yes	Yes	No
Service level support	Yes	No	Yes	No
File attribute support	Yes	Yes	Yes	Yes (z)
Byte range locking	Yes	Yes	Yes	No
ZFILE FSCK support	Yes	Yes	Yes	Yes
TPF4.1 interoperability	No	No	No	Yes
Relative "performance"	2	1	3	4