Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 4, April 2013, pg.256 – 262

RESEARCH ARTICLE

IBM I SERIES OVER IBM Z SERIES

MUPPALLA PRUDHVI[1], KOLA SIVA THARUN[2], SATTI SWAMI REDDY[3]

[1]Electronics and Communication Engineering, KL University, India

[2]Electronics and Communication Engineering, KL University, India

[3]Electrical and Electronics Engineering, KL University, India

[1] Prudhvi.muppalla@gmail.com; [2] tharun200913@gmail.com; [3] swami.neon@gmail.com

Abstract— In this paper we are going to discuss about MI architecture its characteristics and SLIC, how JVM is used to stimulates the computer in software and comparison of IBM I Series over IBM Z series in detail. The AS/400 - formally renamed the "IBM iSeries," but still commonly known as AS/400 - is a midrange server designed for small businesses and departments in large enterprises and now redesigned so that it will work well in distributed networks with Web applications. The AS/400 uses the PowerPC microprocessor with its reduced instruction set computer technology. Its operating system is called the OS/400, with multi-terabytes of disk storage and a Java virtual memory closely tied into the operating system. IBM System z is a family name used by IBM for all of its mainframe computers.

Key Terms: - MI (Machine Interface); JVM (Java Virtual Machine); SLIC (Subscriber Line Interface Circuit); TFS (Temporary File Systems); API (Application Programming Interface)

I. INTRODUCTION

The systems are built with spare components capable of hot failovers to ensure continuous operations; the System z family maintains full backward compatibility. The Z series family, which included the z900, z800, z990 and z890, introduced IBM's newly-designed 64-bit z/Architecture to the mainframe world. The new servers provided more than four times the performance of previous models. In its 64-bit

MI architecture:

The MI architecture is a logical interface between the 2 layers of the software. In which application programs and Os/400 programs reside above the MI and SLIC reside below the MI. The SLIC will insulate the application program and Os/400 program from the underlying hardware characteristics. When changes are made to the hardware SLIC can rewrite the changes. The MI architecture has 2 components One is set of instructions and the other one is operand. These operands are traditional bits and bytes and other is complex data structures called objects which are supported by MI architecture. The traditional computer design represents its information as directories, database files etc and the object at the MI is a container which holds the data structure that represents the information instead of application and operating system.

Theory:

I series is not only the computer design which uses a technology independent machine interface. Then **JVM** is used which simulates the computer in software. The **JVM** is like **MI** which resides between the 2 layers of the software, java application above it and operating system below it. The major difference between the **JVM** and **MI** is the way in which the Instructions are executed. The java compiler produces the instructions at JVM interface called byte codes, which is similar to an **RPG** compiler creating **MI** instructions. The only exception is

MI program contains MI instruction which is translated into lower level instructions before execution of program.

Machine interface characteristics:

1) Conventional machine interface has both op-code and one or more operand fields. The type of operation in this machine is branching, data manipulation etc and this works on registers, memory or immediate data. For example "add register" we have to take the bits in one register, add it in other register and put the result in some place. The problem with this structure is it is a very hardware technology dependent. If there are some changes in the structure then the instructions also changes.
2) The MI architecture also has same type of operands and the instructions are similar to API's format and these API's operate an objects rather than bits in register. In MI there are no registers or immediate data but they have objects. There are several object types in MI.
3) The main object is space. "space" is simply a chunk of bytes.

Programs:

The MI instructions works on the programs where the program is an object. These instructions operate on the program as a complete entity. The instructions will operate on the entire data.

Creating a program:

We create a program from program template which is a predefined structure and the code generation part of the iseries compilers generates this program template. All MI systems are created from templates which are contained in spaces at MI and each object has its own unique template. The create program points to a program template. Here we have system pointer and space pointer which each are 16 bytes long system pointer points to the MI object and space pointer points to the byte in space. The code in the SLIC implements the create program instruction. First space pointer in the create program is used to access the program template then there is a syntax check where the template is correct or not. If it is correct then the translator in SLIC is invoked to transform the MI instruction. Then the address to the create program instruction is send to the object.

Destroying a program:

As the MI system which is created can also be destroyed. There are some instructions to destroy the objects. A user at the MI level says to destroy it then it will be destroyed. But any user can not be destroyed there should be proper authority. But in some cases not only the user but the other users have to destroy it. At that time it will check for "user profile". The user profile among the other things identifies the authorization. When user issues destroy instruction the system first checks the user profile to check whether the user has authority to destroy the particular object before the operation. There are times when the application or operating system software must look at program characteristics. An MI instruction lets a program be "**materialized**". The materialized program instruction points to an encapsulated program object and recreates the program template to that program. Materialization is opposite to encapsulation.

Division of responsibilities:

The OS/400 functions have SLIC implementations. The division between the OS/400 and the SLIC are work management, which schedules the job in iseries, can be in OS/400 because such functions have few hardware dependencies. Other functions such as device support can be partially in SLIC and OS/400. Major operating system functions such as security, have some portions above and below the MI. Even the individual components of an operating system function can be split.

Microcode:

Microprogramming has been described as an implementation technique in which inner computer is programmed to emulate the operations of outer computer architecture. The software to accomplish this is called microcode. The idea behind these systems is to build a hardware that was compatible with IBM's then sell it lower price. If we sold the hardware we lose the technology independence by MI. We need MI as external interface. We need to package a machine product (MP) that contains the hardware plus kernel of the operating system.

Development of SLIC:

There are 2 ways to change the code in MI. One is to rewrite and redesign and next one is to move the affected low level components to the RISC hardware with minimal changes. This type of migration without changing the program logic is called porting. Then they use "OO" programming to improve the productivity.

OO concepts:

An **object** is software which combines the data and the operations that can be performed on it. The operation performed by the object is called "**method**". The data structures and the methods which are hidden by the program is called as **"encapsulation**". **Class** is primary mechanism for reuse. Multiple objects can be created on these classes. This is called as instance of the object. If there is a subclass form the existing class then

it is called as "**inheritance**". The ability of the subclasses of the same class to respond the same input message and produce different result is called "**polymorphism**".

Tools:
The Machine product and PL/MP and OO programming all these are development tools.

Cost of SLIC:
To build a computer system with a high level machine interface and a large portion of operating system below this interface have an associated cost. This increased cost comes from the software development. The SLIC functions work together as a whole. Because all of SLIC is developed under roof, we achieve a level of consistency. Sharing common software parts across multiple operating systems may be nice of saving cost but can't match the integration of function in the iseries.

Comparing with IBM Z series:
The architecture of Z series is **mainframes**. First generation systems, such as the IBM 705 in 1954 and its successor generation, the IBM 1401 in 1959, were a far cry from the enormously powerful and economical machines that were to follow, but they clearly had characteristics of mainframe computers. The IBM 1401 was called the Model T of the computer business, because it was the first mass-produced digital, all-transistorized, business computer that could be afforded by many businesses worldwide. These Computers were sold as business machines and served then, as now, as the central data repository in a corporation's data processing center.

The S/360 was also the first of these computers to use *microcode* to implement many of its machine instructions, as opposed to having all of its machine instructions hardwired into its circuitry. Microcode (or *firmware*) consists of stored microinstructions, not available to users that provide a functional layer between hardware and software. The advantage of microcoding is flexibility, where any correction or new function can be implemented by just changing the existing microcode, rather than replacing the computer.

Architecture is a set of defined terms and rules that are used as instructions to build products. In computer science, architecture describes the organizational structure of a system. Architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, Components and subsystems. Starting with the first large machines, which arrived on the scene in the 1960s and became known as "Big Iron" (in contrast to smaller departmental systems), each new generation of mainframe computers, has included improvements in one or more of the following areas of the architecture:
_ More and faster processors
_ More physical memory and greater memory addressing capability
_ Dynamic capabilities for upgrading both hardware and software
_ Increased automation along with hardware error checking and recovery
_ Enhanced devices for input/output (I/O) and more and faster paths (*channels*)
Between I/O devices and processors

Today's mainframe generation provides a significant increase in system scalability over the previous mainframe servers. With increased performance and total system capacity, customers continue to consolidate diverse applications on a single platform. New innovations help to ensure it is a Security-rich platform that can help maximize the resources and their utilization, and can help provide the ability to integrate applications and data across a single infrastructure. The current mainframe is built using a modular design that supports a packaging concept based on *books*. One to four books can be Configured, each containing a processor housing that hosts the central processor units, memory, and high speed connectors for I/O. This approach enables many of the high-availability, nondisruptive capabilities that differentiate it from other platforms. Mainframes, however, tend to be hidden from the public eye. They do their jobs dependably (indeed, with almost total reliability) and are highly resistant to most forms of insidious abuse that afflict PCs, such as email-borne viruses. By performing stably, quietly, and with negligible downtime, mainframes are the example by which all other computers are judged. But at the same time, this lack of attention tends to allow them to fade into the background. The difference is I series is technology independent machine interface and the Z series is mainframe architecture.

II. FILE SYSTEMS OVERVIEW OF I-SERIES

File system is one of the most visible part of an operating system. Most operating systems are user defined named objects called as "**files**". These files can hold programs, data and whatever the user wants. Then the operating system provides API's to create, destroy, read, write and manage files. Most operating systems have their own distinct file types. Eg: UNIX has files, directories, character stream files and regular files contain user data. Directories are used to keep the track of the file. Integrated file systems are introduced at V3R1 which creates a consistent structure for all the existing files and also create a new file structure needed for other operating system. The IFS support 10 file systems and 3 server types. IFS which are a part of OS/400 provide a consistent structure for the user, and applications to provide all the traditional database files, libraries, folders and other entities. To manage all the files we create a new hierarchical directory structure. This allows access o objects by specifying path through the directories.

Shared folders were introduced in OS/400 and these folders object was added o support these office functions. Mails, documents, programs and files were among the traditional items that could be in the filing system. Document library services let users treat the filling system as an electronic filing cabinet complete with document library object in folders.

The introduction of integrated file system not only creates a consistent structure for all existing file systems on AS/400 but also creates a new file structure for other operating system. The IFS is a part of AS/400 which provides a consistent structure for users to access all traditional libraries, folders documents and other entities and also supports input/output streaming. For managing files a new hierarchical directory structure is used which allows access to objects by specifying the path through the directories to the objects in the similar manner done in personal computer.

Some of the key features of the integrated file system are

1) It will support for storing the information in the file which contain long continuous strings of the data

2) A hierarchy directory structure allows objects to be organized like fruit on the branches of a tree.

3) Not only the stream file allows users applications to access but also data files, documents and other objects that are stored in your server.

4) The common view of the stream files are stored locally on the server, integrated x Series server and also remote windows server. Stream files can also be stored on LAN server or a network file system server.

Some of the reasons why we are using the integrated file system, they are:

1) We can use integrated file system to provide fast access to os/400 server data, client access that uses the OS/400 file server.

2) We can use integrated file system for handling stream data such as images, audio and video more effectively.

3) Provide a file system base and directory base for supporting UNIX based open standards such as portable operating system interface for computer interface. The file structure and directory structure also provides a familiar environment for users operating systems such as disk operating systems and windows 95/98 and NT.

There are different file systems which an IFS supports. They are:

1) **Root file system**: The root file system contains the files and directories of the disk operating system and windows file system. It is accessed through a hierarchical directory structure similar to unix system and is optimized for stream file input and output.

2**) Open system file system**: The open file system contains UNIX based files and directories.

a**) Library file system**: The library file system is the library structure of AS/400. It contains libraries and other type of objects. The QSYS.LIB file system provides access to these objects in a hierarchical directory structure for user. It supports the creation of libraries, source physical files; data and source physical file members, and user spaces. The ability to delete, renames, move, copy, and retrieve attributes for most objects that reside in a library is also supported.

b) **Document library services file systems:** The QDLS file system supports the folders structure originally added to OS/400 from the S/36. It provides access to documents and folders. In addition it supports OS/400 folders, document library objects, and data stored in stream files.

c) **Optical file system**: the QOPT **file** system provides access to stream data that is stored on optical media. It too is optimized for stream file I/O and supports data stored in stream files.

d**) OS/400 file server file system**: This file system provides access to other file systems that reside only on remote iseries or AS/400 servers. The file system acts as a client on behalf of users to perform file requests. This interacts with file server on the target system to perform the actual file operation.

3) **User defined file systems**. A user-defined file system contains user-defined and user-management file system. UDFS resides in an auxiliary storage pool (ASP).

a) Network File System: A Network File System gives users access to data and objects that are stored on a remote NFS server.

IFS also supports for servers

1) NFS server: The NFS server and NFS file system let the iseries act as both client and server by allowing remote file systems or directories. This make the iSeries distribute data access a network by exporting local file systems for remote clients to access.

2) OS/400 Net Server: This windows PC's access IFS directories and OS/400 output queues. So that we need TCP/IP configured on both the iseries and PC.

3) OS/400 Remote file system: This RFS lets a client transparently access file systems that reside on remote iseries or AS/400 servers. This is accessed through the hierarchical directory structure with a first level directory. RFS provides a fairly simple and efficient way to share files between two iseries or AS/400 servers.

Conclusion

IFS is a fundamental application enabler for the iseries which enhances the already extensive data management capabilities of OS/400 extends for new applications

Now I am **comparing** with **IBM Z-series.**

Z-series file systems over-view:

The Z-series file systems is a unix system services, file system that can be used in addition to the hierarchical file system. ZFS file systems contains files and directories that can be accessed with Z/OS UNIX application programming interface (API's). These file systems can support access control lists (ACLs). zFS file systems can be mounted into the z/OS UNIX hierarchy along with other local (or remote) file system types (for example, HFS, TFS, AUTOMNT and NFS).

zFS does not replace HFS, rather zFS is complementary to HFS. zFS can be used for all levels of the z/OS UNIX System Services hierarchy (including the root file system) when all members are at the z/OS V1R7 level. Because zFS has higher performance characteristics than HFS and is the strategic file system, HFS might not be supported in future releases, which will cause you to migrate the remaining HFS file systems to zFS. zFS has the capability of running sysplex aware for read-write mounted file systems. As in previous releases, zFS continues to support running sysplex-aware for read-only mounted file systems. zFS and HFS can both participate in a shared sysplex. However, only zFS supports security labels. Therefore, in a multilevel-secure environment, you must use zFS file systems instead of HFS file systems. Multi-file system aggregate support is not planned to be enhanced and might be removed sometime in the future. IBM recommends that you use zFS compatibility mode aggregates rather than zFS multi-file system aggregates. If you have any data stored in zFS multi-file system aggregates, copy that data from the zFS multi-file system aggregate file systems into zFS compatibility mode aggregates. zFS multi-file system aggregates cannot be attached in a shared file system environment. You must copy the data from any file systems contained in multi-file system aggregates into zFS compatibility mode file systems using a non-shared file system environment. zFS supports **a shared file** system capability in a multisystem sysplex environment. The term ***shared file system environment*** means, users in a sysplex can access zFS data that is ***owned*** by another system in the sysplex. For full sysplex support, zFS must be running on all systems in the sysplex in a shared file system environment and all zFS file systems must be compatibility mode file systems (that is, they cannot

be file systems in multi-file system aggregates). zFS multi-file system aggregates are not supported by auto mount. In ZFS, file system manipulation within a storage pool is easier than volume manipulation within a traditional file system; the time and effort required to create or resize a ZFS file system is closer to that of making a new directory than it is to volume manipulation in some other systems.

Features of ZFS:

There are many features provided by ZFS. They are

Performance: ZFS provide better performance in many of the cases. This provides a good improvement in running in a shared file system environment.

Restart: ZFS reduces the loss of updates. ZFS is a logging file system. It logs Meta data updates. If some failure is occurred in a system, the ZFS replaces the log to make the file system consistent.

Cloning: ZFS makes the clone read only to the user which is a same data set in a file system. This clone file system can be made available to the users to provide a copy of the file.

III. COMPARING ZSF WITH IBM SYSTEM I

z/OS UNIX System Services (z/OS UNIX) allows z/OS users to create UNIX file systems and file system directory trees on z/OS, and to access UNIX files on z/OS and other systems. In z/OS, a UNIX file system is mounted over an empty directory by the system programmer (or a user with mount authority).

You can use the following file system types with z/OS UNIX:

1) **System z File System (**zFS), which is a file system that stores files in VSAM linear data sets.

2) **Hierarchical file system** (HFS), a mountable file system, which is being phased out by zFS.

3)z/OS Network File System (z/OS NFS), which allows a z/OS system to access a remote UNIX (z/OS or non-z/OS) file system over TCP/IP, as though it were part of the local z/OS directory tree.

4) **Temporary file system** (TFS), which is a temporary, in-memory physical file system that supports in-storage mountable file systems. As with other UNIX file systems, a path name identifies a file and consists of directory names and a file name. A fully qualified file name, which consists of the name of each directory in the path to a file plus the file name itself, can be up to 1023 bytes long.

The z/OS Distributed File Service (DFS) System z File System (zFS) is a z/OS UNIX System Services (z/OS UNIX) file system that can be used in addition to the hierarchical file system (HFS). zFS file systems contain files and directories that can be accessed with z/OS UNIX application programming interfaces(APIs). These file systems can support access control lists (ACLs). zFS file systems can be mounted into the z/OS UNIX hierarchy along with other local (or remote) file system types .where as in IBM system I we have IFS which is integrated file system and it deals with the files ". These files can hold programs, data and whatever the user wants. Then the operating system provides API's to create, destroy, read, write and manage files. Most operating systems have their own distinct file types. Eg: UNIX has files, directories, character stream files and regular files contain user data. Directories are used to keep the track of the file. Integrated file systems are introduced at V3R1 which creates a consistent structure for all the existing files and also create a new file structure needed for other operating system. The IFS support 10 file systems and 3 server types. IFS which are a part of OS/400 provide a consistent structure for the user, and applications to provide all the traditional database files, libraries, folders and other entities. In IBM iSeries it deals with integrated file system where as in ZFS it deals with a logging file system. It logs Meta data updates. If some failure is occurred in a system, the ZFS replaces the log to make the file system consistent. In **iSeries the IFS** has **shared folders** in the file system where as in the **ZFS** it has **shared file system** environment. We have different file systems in IFS such as

1) root file system
2) Open file system
3) User defined file system

We have different file systems in ZFS also. They are

1) System Z file system
2) Hierarchy file system
3) Temporary file system

The z/OS Distributed File Service (DFS) System z File System (zFS) is a z/OS UNIX System Services (z/OS UNIX) file system that can be used in addition to the hierarchical file system (HFS) where as in IBM iSeries we use OS/400 server. Both I series and z series contains files and directories in the file system and operations are made on these file systems.

IV. CONCLUSION

The technology independence that MI provides is important because it doesn't change the user application. New hardware can be added and exploited. . New instructions and functions are added and the MI can support API's. The MI is a powerful interface not only because of technology independent but because of its expandability MI has long life ahead. IFS is a fundamental application enabler for the iseries which enhances the already extensive data management capabilities of OS/400 extends for new applications.

ACKNOWLEDGEMENT

First I am grateful to God for giving me a chance to complete this work. This work is supported by J. Avinash (Assistant Professor) of ECM Department, K L University. I am thankful to him and K L University for giving me continuous encouragement and support.

REFERENCES

[1] Bill white,Mario Almedia and Dick Jorna , IBM @eserver zSeries 990 Technical Guide,Redbooks ,Second Ed.,May 2004.

[2] Paul Rogers and Robert Hering ,z/OS Distributed File Service zSeries File System Implementation z/OS V1R13,Red books,Sixth Ed.,Oct.2012.

[3] MileMaker AS/400 On Demand (iSeries) Technical Guide,RAND MCNALLY.
[4] Mike Ebbers,John Kettner ,Wayne O'Brien and Bill Ogden, Introduction to the New Mainframe z/OS Basics,Redbooks,Third Ed.,Mar.2011
[5] iSeries Integrated File System Version 5 Release 3,Sixth Ed.,IBM,Aug.2005.
[6] IBM Systems - iSeries Hierarchical File System APIs Version 5 Release 4,IBM,Sixth Ed.,Feb.2006.
[7] Jim Hoskins and Bob Frank,Exploring IBM eserver zSeries and S/390 Servers,MAXIMUM PRESS,Eight Ed.
[8] Bill Venners, Inside the JAVA Virtual Machine,McGraw-Hill Osborne Media ,Second Ed.,2000.
[9] Louis Scheffer, Luciano Lavagno and Grant Martin ,EDA for IC System Design,Verification , and Testing ,CRC Press, 2006.

