Scalability and High Availability

Part 2: z/OS Operating System

Robert Vaupel
IBM Deutschland Research & Development GmbH
vaupel@de.ibm.com
## Scalability and High Availability

<table>
<thead>
<tr>
<th>Date (Fridays)</th>
<th>11:30-13:00</th>
<th>14:00-15:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.12.2010</td>
<td><strong>z/OS Introduction</strong>&lt;br&gt;History and Basic Concepts Program Execution and Recovery, Data Exchange, I/O</td>
<td><strong>Using z/OS</strong>&lt;br&gt;TSO, ISPF and Job Control Demo</td>
</tr>
<tr>
<td>17.12.2010</td>
<td><strong>Dispatching and Virtualization</strong>&lt;br&gt;z/OS and PR/SM</td>
<td><strong>Dispatching and Virtualization</strong>&lt;br&gt;Locking and Hiperdispatch</td>
</tr>
<tr>
<td>14.1.2011</td>
<td><strong>Parallel Sysplex</strong>&lt;br&gt;Local and Global Clusters</td>
<td><strong>z/OS Workload Management</strong>&lt;br&gt;Basic Concepts and Algorithms</td>
</tr>
<tr>
<td>28.1.2011</td>
<td><strong>z/OS Workload Management</strong>&lt;br&gt;Advanced Concepts and Middleware Integration</td>
<td><strong>Software Architecture on System z</strong></td>
</tr>
</tbody>
</table>
Economical Importance: Why System z and z/OS

- All companies which have the need to store huge amounts of data require
  - Security
  - Scalability
  - Compatibility
  - Availability
  - Reliability
  - Serviceability

- 95% of the 2000 world-wide biggest companies use System z computers
- Around 65-70% of all relevant data are stored on System z computers
- 60% of all data being access thru the world wide web are stored in databases on System z (DB2, VSAM, and IMS)
Introduction of S/360 Architecture

7. April 1964

System z and z/OS History

1960
1965
1970
1975
1980
1985
1990
1995
2000
2005

S/360
S/370
S/390
z Architecture

MVT
MFT

SVS

MVS/370

MVS/XA

MVS/ESA

OS/390

z/OS

Fixed Storage
15 Partitions or Tasks

Address Spaces
Multiple Virtual Storage

2 GB Virtual Storage

Expanded Storage
Virtual I/O
Fast Program Load
Dynamic I/O
Posix Cluster

Parallel Sysplex
Workload Management
Unix System Services
TCP/IP ...

Java
Websphere
IEEE Float
64 bit
IRD
Hiperdispatch
Offload
Security
GDPS ...

Symmetric Multi Processing
Virtual Memory

Expanded Storage
2GB Addressing LPAR

Access Registers
Data-spaces

CMOS Technology
Parallel Sysplex

64bit Addressing

 iterative would
Introduction to z/OS

- z/OS Structure and Size
- Address Space Concept
- Program Execution
  - Program Recovery
  - Data Exchange
- z/OS Data Sets and I/O Flow
- Major Subsystems
  - Job Entry System
  - Time Sharing Option
  - (Unix System Services)
- Demo: TSO, ISPF and JES
- z/OS IPL
z/OS Structure

- Basic Control Program (~System Kernel)
  - Supervisor/Dispatcher
  - Console Services
  - Recovery
  - Storage Management
  - Unix System Services
  - IPL
  - Resource and Workload Management
  - Sysplex Communication

- Base Elements
  - Elements which are required to run the system
    - DFSMS (I/O storage management)
    - Installation
    - Unix System Services
    - Language Environments
    - Job Entry System

- Optional Features
  - Monitors and Add-on Features
  - Features which can be replaced by vendor products

- System doesn’t run without BCP+Base Elements
  - Optional Features are required

- On-top
  - Middleware: CICS, IMS, DB2, WAS, MQ
  - Customer Applications
  - Vendor Applications
z/OS in Numbers

- Size of z/OS ~105 Million Lines of Code (LOC)
- Test code ~740 Million LOC
- Tools (internal+IBM wide) ~10 Million LOC
- Altogether ~ 850 Million LOC
Address Spaces

- Address Spaces encapsulate users of the operating system and provide
  - Virtual Storage
    - 16MB → 2GB → 18EB
  - Structure to execute programs
    - Tasks
    - Program Status Word (PSW)
    - Control Registers
    - General Purpose Registers (GPR)
    - Access Registers (AR)
    - Floating Point Registers
Address Spaces: Storage Layout

- Provide virtual storage map for users and applications
  - Fences application data
  - Base concept to assure data integrity

- Grew in size over time
  - • 16MB, 370 architecture
  - • 2 GB, XA extended architecture
  - -> 2 GB, z architecture

- Areas
  - Private area
    - Application data
    - Local control data
  - Common Area
    - Operating system data
    - Data which needs to be shared between applications
      - Authorized applications can access the common area
Common Storage

- Communication Vector Table (CVT)
  - Anchor to all system control structures
    - Also for important subsystems and add-on components
  - Access requires Supervisor State and Key=0
  - Base address is at fixed location X’10’ in prefixed save area (PSA)

- Prefixed Save Area
  - Nucleus (z/OS equivalent to kernel) storage area with fixed addresses
  - Loaded by IPL time
  - Anchors to all system relevant structures
Address Space: Creation

- **Start Command**
  - Creates an Address Space (Memory Create)
  - Creates Region Control Task (RCT)

- **Region Control Task**
  - Creates Started Task Control (STC) and INIT Task

- **Started Task Control**
  - Controls the conversion of the start procedure
  - Receives the job id
  - Links to INIT task

- **INIT Task – Interpreter**
  - Interprets the start procedure
  - Creates data and allocates devices
  - Creates Job/Step Task and loads the program which should be executed

- **Job/Step Task**
  - Executes the program

- **For Batch jobs**
  - Address space already exists up to INIT task
    - Only Interpretation of Job control and attaching the J/S task is required
Execution Units and Programs

- **Program Execution**
  - Requires Control Structures
    - TCB – Task Control Block
      - Executes User Programs
    - SRB – Service Request Block
      - Executes System Programs on behalf of user requests
      - SVC – Supervisor Call: Invokes System Services
  - TCBs and SRBs are execution units which will be placed on the Dispatcher Queues

- **Programs**
  - Are loaded from libraries
  - Standard search for libraries
    - But can be overridden when a new program is started
Recovery: Setup

- All application programs should implement a recovery procedure
- All system programs setup recovery procedures
- Goal
  - Recover from failure and continue processing if possible
  - First Failure Data Capture
  - Ensure that the environment is able to continue processing

- Motivation
  - Standalone program
    - If it abends usually not much damage
  - Server program
    - Handles for example 500 requests from 1000 users in parallel
    - If it abends without recovery all users would terminate
Recovery: Error Processing

- **First failure data capture**
  - Record error
    - SDWA (System Diagnostic Work Area)
    - Initiate dump
    - Log error

- **Repair**
  - Broken control block chains
  - Update data to ensure that the environment is not affected

- **Retry**
  - Possible for anticipated errors
  - Retry footprints and return labels are defined

- **Percolate**
  - Step up to the next error recovery routine
Recovery: Routines

- Problem state programs
  - These are programs which have limited access to system resources
    - For example: restrictions concerning storage access, program invocation
  - Program recovery must be in the same scope
    - ESTAE, ESPIE and associated routines which are only capable to analyze the environment of the application

- Supervisor state programs
  - Have access to nearly everything
  - These programs can use system locks and they can ran disabled
  - Error recovery
    - FRR is an enabled unlock task which protects code that is not disabled, locked and in system mode
Why does a program terminate?

- Typically, a program gets control, performs its function, and terminates normally (via SVC 3, EXIT)

- However, there are ways a program can terminate abnormally
  - Program checks
  - System and application detected errors resulting in ABEND macro calls

- Program Checks
  - Application logic error
  - System-detected software errors (incorrect value specified to a system service)
  - Hardware-detected errors

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0C1</td>
<td>(Privileged) Operation Exception</td>
</tr>
<tr>
<td>0C2</td>
<td>Invalid instruction, read to not closed file, invalid subroutine call …</td>
</tr>
<tr>
<td>0C4-4</td>
<td>Protection Exception: Invalid storage key</td>
</tr>
<tr>
<td>0C4-10</td>
<td>Segment Table Translation Exception: segment table not loaded (storage not obtained)</td>
</tr>
<tr>
<td>0C4-11</td>
<td>Page Table Translation Exception: page table does not exist (storage not obtained)</td>
</tr>
<tr>
<td>0C5</td>
<td>Addressing Exception: main storage location is not available in the configuration</td>
</tr>
<tr>
<td>0C7</td>
<td>Various data exceptions</td>
</tr>
<tr>
<td>0C9</td>
<td>Fixed point divide exception</td>
</tr>
</tbody>
</table>
Programs: Data Exchange

- Problem
  - Within an Address Space
    - All data can be accessed
      - Just Common Storage requires program authorization
  - Address Spaces fence data and programs
    - How is it possible for programs to exchange data between address spaces?

- 3 “classical” techniques
  - Via Common Storage
  - Via Cross Memory
  - By directly addressing the data in the other address space

- Also with Unix System Services
  - Via Shared Memory Objects (not discussed)
What to do?
- Program A wants to copy data into Address Space B

Process
1. Program A copies the data into common storage
2. Program A schedules an SRB to Address Space B
3. The SRB is able to copy the data from common storage to local storage for address space B

Remarks
- Oldest technique
- Requires authorization to access common storage
  - Useful to move/copy small amounts of data to address spaces with lower authorization
- Common storage is a limited resource
What to do?
- Program A wants to copy data into Address Space B [or execute a function in another address space]

Process
1. Program A calls a program of another address space
   - Home AS: originating address space
   - Primary AS: current execution AS
   - Secondary AS: target execution AS
2. Home, primary and secondary AS is available to each program via control registers
   - Called program runs in primary address space
   - MVCP copies data from secondary to primary address space
3. MVCS allows to copy data into the other direction

Remarks
- Efficient methods for applications or subsystems which offer service to other applications
  - Often used by operating system components
What to do?
- Program A wants to copy data into Address Space B

Process
1. Program A needs to know the address space number of the target address space
   - SSAR: Set Secondary Address Space allows to modify Control Register 3 which keeps the Secondary ASN
2. Program A also needs to know an access list entry for the target address space
   - To each GPR exists an AR which specifies to which address space the address in the GPR belongs to
   - For secondary address spaces the ALET is always 2
3. MVCP copies data from secondary (target to primary)
4. MVCS allows to copy data into the other direction

Remarks
- Allows direct access to local data of other address spaces
- Requires authorization
- Efficient way for operating system and middleware
z/OS Data Sets

- **Extended Count Key Data (ECKD) Architecture**
  - Data sets are a collection of records
  - Each record contains a count, a key and the data

- **Data Storage: Direct Access Storage Devices (DASD)**
  - Geometry: Cylinder, Tracks, Records
  - Address of a data set:
    - Combination of cylinder, head (track) and record: cchhr
    - 01023 ... 3rd record on cylinder 1 and track 2
  - Today
    - Only logical volumes exist
    - No physical /390 disks exist anymore
    - Logical volumes can go up to 54GB
    - With Super Large Volume support nearly unlimited

- **z/OS**
  - Allows for $2^{16} = 65536$ device addresses
  - This is extended by multiple sub-channel sets but each device needs at least one address within the 1st sub-channel set
z/OS Data Sets

- Data set is a collection of single files
  - Originally: VSAM (Virtual Storage Access Method) consisted of at least 2 files an index and a data file

- Data set types
  - Sequential
    - Fixed or variable length
  - Partitioned
    - Consists of a directory and multiple members
      - Each member looks like a single file
    - Used for program libraries
  - VSAM
    - Multiple organizational formats
    - Base for data bases on MVS
  - HFS Hierarchical File System and zFS
    - UNIX file systems on z/OS
z/OS Data Sets: Organization

Master Catalog
- User → User catalog
- User (Alias) → User

User (ICF) Catalog
- User
  - List of datasets with volumes

DASD (Volume, VOLSER)
- VTOC
  - Physical Address of data set
  - Data sets

VTOC – Volume Table of Content
DASD – Direct Access Storage
VOLSER – Volume and Serial Number
z/OS I/O Flow

1. Program
2. Access Method
3. EXCP
4. IOSB
5. CCW
6. CCW
7. I/O Interrupt
8. ECB
9. wait
10. Program

I/O Driver

UCB 2000B
UCB 2000A
IOQ

I/O Supervisor

Channel Subsystem

S/C 13FE
S/C 13FD
S/C 13FC

Device

2000A
z/OS I/O Flow …

1. Application program issues Open macro and tells system which dataset it wants to access. OPEN tests access rights and locks dataset access. Now the application program can use PUT, GET, READ, or WRITE macros to access the dataset.

2. The access method create the channel program which contains of a sequence of channel command words. It also implements data buffering and synchronization and is able to re-initiate the I/O operation in case of an error.

3. The access method calls the I/O driver (usually EXCP) in order to move the data. The I/O driver translates the virtual addresses of the channel program into real addresses. For that reason the I/O driver enters supervisor state. Also the I/O driver can reserve pages in real storage to which the channel subsystem can copy data to or from.

4. The I/O Supervisor is called and the channel program is started. The channel program is queued to a Unit Control Block which represents the device in the system. If the UCB is already in use by another channel program the current program is queued. Otherwise a start subchannel command is issued to give the channel program to the Channel Subsystem. In any case the processor is given up now and the dispatcher can schedule a different program to execute.

5. A SAP executes the Start Subchannel and selects a channel to access the control unit and the device. The SAP and the channels use the control structures of the microcode which are the subchannels and which are the device representation within the channel subsystem. For each device a subchannel exists and resides in a hardware area designated for the hardware. The channels now executes the channel program and supervises the movement of data. The first channel initiates the data transfer to the the I/O device. When data is transferred back to the system or a completion is send a different channel can execute it and controls the data transfer backwards.

6. The channel signals the SAP when the I/O Operation is completed. The SAP initiates the I/O Interrupt to the operating system.

7. A I/O supervisor routine executes the I/O interrupt and resumes the waiting task which initiated the I/O operation. Then the control is handed over to the dispatcher.

8. Eventually the dispatcher selects the originating task for execution which resumes the access method.

9. The access method checks the result of the I/O operation and provides its status back to the originating application program.

10. The application resumes its processing.
z/OS Subsystems: Job Entry System

- **Tasks**
  - Controls the correct establishment of externally started tasks
  - Controls the execution of batch jobs in a system and a sysplex
  - Allocates all required resources (devices, runtime specifics) for the execution of the programs
  - Guarantees that no data or jobs get lost

- **JES2**
  - Originally designed for small environments (single systems)
  - Today the dominating job entry system which supports also large sysplex environments
    - *Multi Access Spool: All systems are equal within the “MAS complex”*

- **JES3**
  - Designed for multi-system execution
  - Master/Slave design

```
EDIT ---- userid.SORT.JCL -------------------------- LINE 00000000 COL 001 080
COMMAND ===> SUBMIT SCROLL ===> CSR

**********************************************************************************
000100 //MYUSER01 JOB (ITSO),"IEFBR14",CLASS=A,MSGCLASS=X
000200 //*------This is a comment line -------------------------------------------
000300 //STEP1 EXEC PGM=IEFBR14
000400 //NEWDD DD DSN=MYUSER.IEFBR14.TEST.NEWDD,
000500 // DISP=(NEW,CATLG,DELETE),
000600 // UNIT=SYSDA,
000700 // SPACE=(CYL, (10,10,45)),
000800 // LRECL=80,
000900 // BLKSIZE=3120
```
z/OS Subsystems: Job Entry System

Job

Input

Conversion Queue

Processing Queue

Output Queue

Hardcopy Queue

Purge Queue

SYSIN

SYSOUT

JCL

Job Control Language (JCL)

Input data (SYSIN)

SPOOL

SPOOL = Simultaneous Peripheral Operations Online

Other Outputs

© 2010 IBM Corporation
z/OS Subsystems: Time Sharing Option

- **Time Sharing Option (TSO)**
  - Subsystem for inter-active users
  - Access via network (VTAM = Virtual Terminal Access Method)

- **Usage**
  - Originally it was used by everybody: developers, system programmers, office worker, secretary, etc...
  - Today it is mostly used by system programmers and operators for controlling and maintaining the system

- **Important add-ons:**
  - ISPF: Interactive System Product Facility (user interface)
  - REXX: Restructured Extended Execution Language (command language)
TSO LOGON Screen

Enter LOGON parameters below:

Userid ==> DAD
Password ==> [redacted]
Procedure ==> BASIC
Acct Nmbr ==> DE03141
Size ==> 100000
Perform ==> 
Command ==> 

Enter an 'S' before each option desired below:
-Monmail -Nonotice -Reconnect -OIDcard

PF1/PF13 ==> Help  PF3/PF15 ==> Logoff  PA1 ==> Attention  PA2 ==> Reshow

You may request specific help information by entering a '?' in any entry field.

Connected to remote server/host tn3270.de.ibm.com using lu/pr
TSO: LOGON Procedure

Procedure ==> BASIC

//BASIC    PROC
//***************************************************************
//* TSO BASIC LOGON PROCEDURE
//***************************************************************
//BASIC    EXEC
PGM=Ikjeft01,DYNAMNBR=750,REGION=0M,TIME=1440,
// PARM="EXEC '"SYS4.SYSPROC(LOGON)"'
//*
//SYSPRINT DD TERM=TS,SYSSOUT=* 
//SYSTERM  DD TERM=TS,SYSSOUT=* 
//SYSIN    DD TERM=TS

Environment Variables
System Specific processing
Allocates Default Concatenation
Allocates Default ISPF Environment
Invokes “Command Lists”
• SYS4.SYSPROC(ISPF)
• SYS4.SYSPROC(LOCAL)

Standard Setup

Location Specific Setup

User Specific Setup

User specifications and allocations
z/OS Subsystems: TSO: ISPF

ISPF Main Panel
z/OS Subsystems: TSO: REXX

```
EDIT Command ==> BVAU.MVS.CLIST(HELLOW) - 01.00

******* ********************************************************************
Command ==> BVAU.MVS.CLIST(HELLOW) - 01.00
00001 /** REXX  ******************************************************/
000012 /* Just prints Hello World on the TSO User terminal */
000003 /** hello world on the TSO User terminal **************/
000004 Parse Upper Arg Inps
000005 If Inps = "" then say "Hello World"
000006 else say Inps
000007 Exit

******* ********************************************************************
```

```
EDIT Command ==> BVAU.MVS.CLIST

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Prompt</th>
<th>Size</th>
<th>Created</th>
<th>Changed</th>
<th>TI</th>
</tr>
</thead>
</table>
| 1 | HELLOW    | *RC=0  | 7    | 2003/07/10 12:29:17 | 2003/07/10 12:29:17 | 04:00-
| 2 | RMHVS     |        | 3    | 1995/04/07 13:56:55 | 1995/04/07 13:56:55 | 04:00-
| 3 | RMI       |        | 3    | 1995/04/07 13:56:55 | 1995/04/07 13:56:55 | 04:00-
| 4 | RMPMVS    |        | 2    | 1999/07/07 17:22:37 | 1999/07/07 17:22:37 | 04:00-
| 5 | RFINF     |        | 107  | 2004/09/07 15:42:07 | 2004/09/07 15:42:07 | 04:00-
| 6 | RTESTRUN  |        | 16   | 2004/10/07 16:04:09 | 2004/10/07 16:04:09 | 04:00-
```

```
Hello World
***
```
z/OS Subsystems: Unix System Services

- USS provides a complete UNIX environment under z/OS
  - POSIX 1003.2
  - XPG/4: X/Open Portability Guides
  - OMVS Kernel is part of z/OS
  - USS user can select between different command shells
    - telnet, rlogin, TSO IShell
    - TSO IShell is ISPF based
  - File Systems
    - HFS and zFS
    - NFS: Network File System
    - DFS: Distributed File System (part of DCE)
z/OS Start: IPL = Initial Program Load

LOADPARM

IODF cuuu LOADxx IMSI AltNuc

0206 01 M 1

SYS1.IPLPARM

LOAD01

SYS1.IODEF

01

SYS1.MVSVM

SYSCALLCAT WMLLIB1 SYS1.CATALOG

SYSCPARM 01

SYS1.LINKLIST

SYS1.LINKLIST

SYS1.NUCLEUS

RIMs IEANUC01 IEANUC05

SYS1.LPALIST

SYS1.LPALIST

IEASYS01

IEASYM01

COMMNDxx

CONSOLxx

LNKLISTxx

LPALSTxx

IEAAPFxx

Device Address 0206

SYS1.IODEF

WLMLIB1

SYS1.CATALOG

link.library

lpalibrary
z/OS IPL: Anchors and Definitions

**LOADPARM**

<table>
<thead>
<tr>
<th>IODF cuuu</th>
<th>LOADxx</th>
<th>IMSi</th>
<th>AltNuc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0206</td>
<td>01</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

- **IPL Console**
  - Address of SYSRES
    - Contains system data sets
  - Definition of LOADPARM
    - Address of IODF
      - I/O Definition File
    - Suffix of LOADxx member
      - Anchor for system configuration
    - Number of Nucleus
      - Practical use only for z/OS development

- **LOADxx Member**
  - Defines Configuration
  - Suffix of IODF
  - Place and name of system catalog
  - Suffix of parameter file (IEASYSxx)
  - Suffix of symbol file (IEASYMxx)

```
SYS1.IPLPARM
LOAD01

IODF  01  SYS1  MVSVM
SYSCAT WLMLIB1 SYS1.CATALOG
SYSPARM 01
SYSPLEX WLMPLEX
IEASYM  (01,L)
```
Example for Parameter files
- All are members of SYS1.PARMLIB
  - IEASYS
    * Anchor for all other parameters files
  - LNKLST
    * Data set concatenation for programs
      → Is like PATH statement in Windows
  - LPALST
    * Data set concatenation for programs which should be loaded
      → Similar to dlls in Windows
  - CONSOL
    * List of all consoles connected to the system
  - COMMND
    * Contains start commands for applications to get started directly after IPL
      → Similar to Autostart folder
- And many many more …
z/OS IPL: Flow

- **Step 1: Initialization**
  - Only 1 processor is enabled
  - Load of I/O Configuration
  - Load of Resource Initialization Modules (RIM) which create system control structures
  - Load of nucleus
  - Start of master address space

- **Start 2: System Start**
  - Activate all processors
  - Switch to master console
    - Step 1 only showed a limited NIP (Nucleus Initialization) console
  - Start of system address spaces
    - Many system functions have their own address spaces
    - These are started and initialized by the master address space
  - Start of Job Entry System
    - Now z/OS BCP (Basic Control Program) is completely initialized and ready

- **Start 3: Start of Subsystems and Applications**
z/OS After IPL

RESPONSE=WLM
I EE1141  11. 11. 47  2001. 302 ACTI VI TY  282
JOBS   M S   TS USERS   SYSAS   I NI TS   ACTI VE/ MAX   VTAM   QAS
00002   00017   00003    00026   00023    00003/00030    00007
LLA     LLA     NSW S   VLF   VLF   VLF   NSW S
QAM     QAM     I EFPRCC NSW S   RACF   RACF   RACF NSW S
APPC    APPC    APPC    NSW S   ASCH   ASCH   ASCH NSW S
ENQMDN  ENQMDN  ENQMDN  OM T S   I XFP   I XFP   I XFP NSW S
JES2    JES2    I EFPRCC NSW S   VTAM   VTAM   VTAM NSW S
TSO     TSO     STEP1   OM T S   RMF    RMF    RMF NSW S
RMFGAT  RMFGAT  I EFPRCC NSW S   TCPI P   TCPI P   TCPI P NSW S
FTP01   FTP01   FTP0E    OM T AO FTP0E1  STEP1 FTP0E OM T AO
M/SNFSS M/SNFSS GFSAM IN NSW S   PORTMAP   PORTMAP   PORTMAP OW T AO
VM      VM      VM      NSW S   00024K - 00088K
BHHB    OM T    DAD    OM T    BVAU    I N

Dispatcher
System Resource Manager
I/O Supervisor
Recovery Termination Manager
Basic Control Programm (BCP)
Licensed Internal Code (LIC)
Hardware
Literature

- Introduction to the New Mainframe: Large-Scale Commercial Computing

- ABCs of z/OS System Programming Volume 11,

- Documents for Workload Management
    - z/OS Workload Manager: How It Works and How To Use It, April 2004
      - Adaptive algorithms for managing a distributed data processing workload

### Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

<table>
<thead>
<tr>
<th>Trademark</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPN*</td>
<td>HiperSockets</td>
</tr>
<tr>
<td>CICS*</td>
<td>HyperSwap</td>
</tr>
<tr>
<td>DB2*</td>
<td>IBM*</td>
</tr>
<tr>
<td>DB2 Connect</td>
<td>IBM eServer</td>
</tr>
<tr>
<td>DirMaint</td>
<td>IBM e(logo)server*</td>
</tr>
<tr>
<td>e-business logo*</td>
<td>IBM logo*</td>
</tr>
<tr>
<td>ECKD</td>
<td>IMS</td>
</tr>
<tr>
<td>Enterprise Storage Server*</td>
<td>Language Environment*</td>
</tr>
<tr>
<td>ESCON*</td>
<td>MQSeries*</td>
</tr>
<tr>
<td>FICON*</td>
<td>Multiprise*</td>
</tr>
<tr>
<td>GDPS*</td>
<td>NetView*</td>
</tr>
<tr>
<td>Geographically Dispersed Parallel Sysplex</td>
<td>On demand business logo</td>
</tr>
</tbody>
</table>

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

Red Hat, the Red Hat "Shadow Man" logo, and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc., in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

**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.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

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.