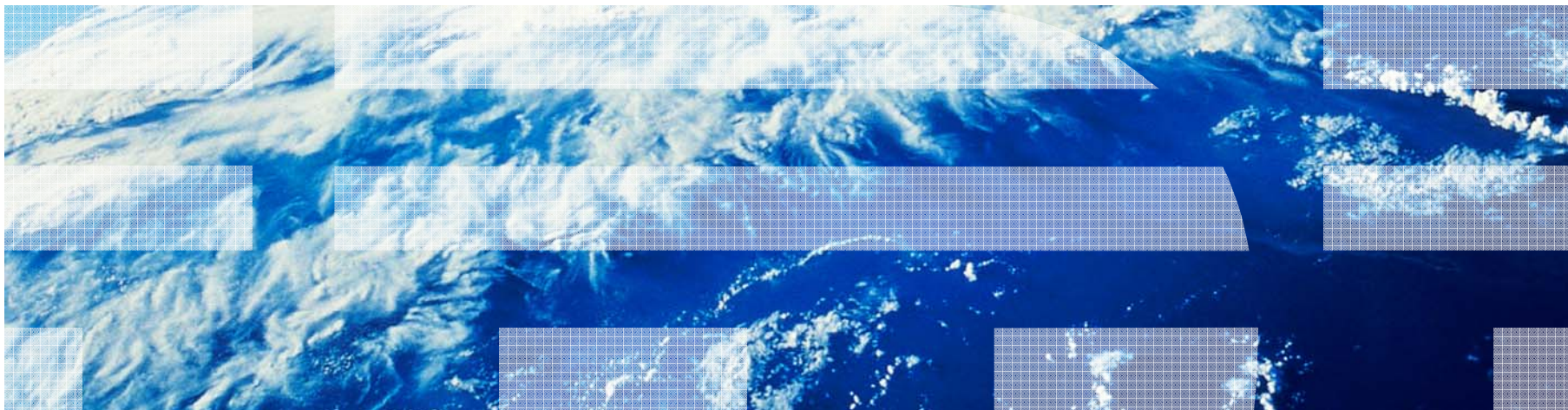


Performance aspects for z/VSE applications with Linux in System z Parallel Access Volumes (PAV)

Ingo Franzki, IBM



Trademarks

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

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business (logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* 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.

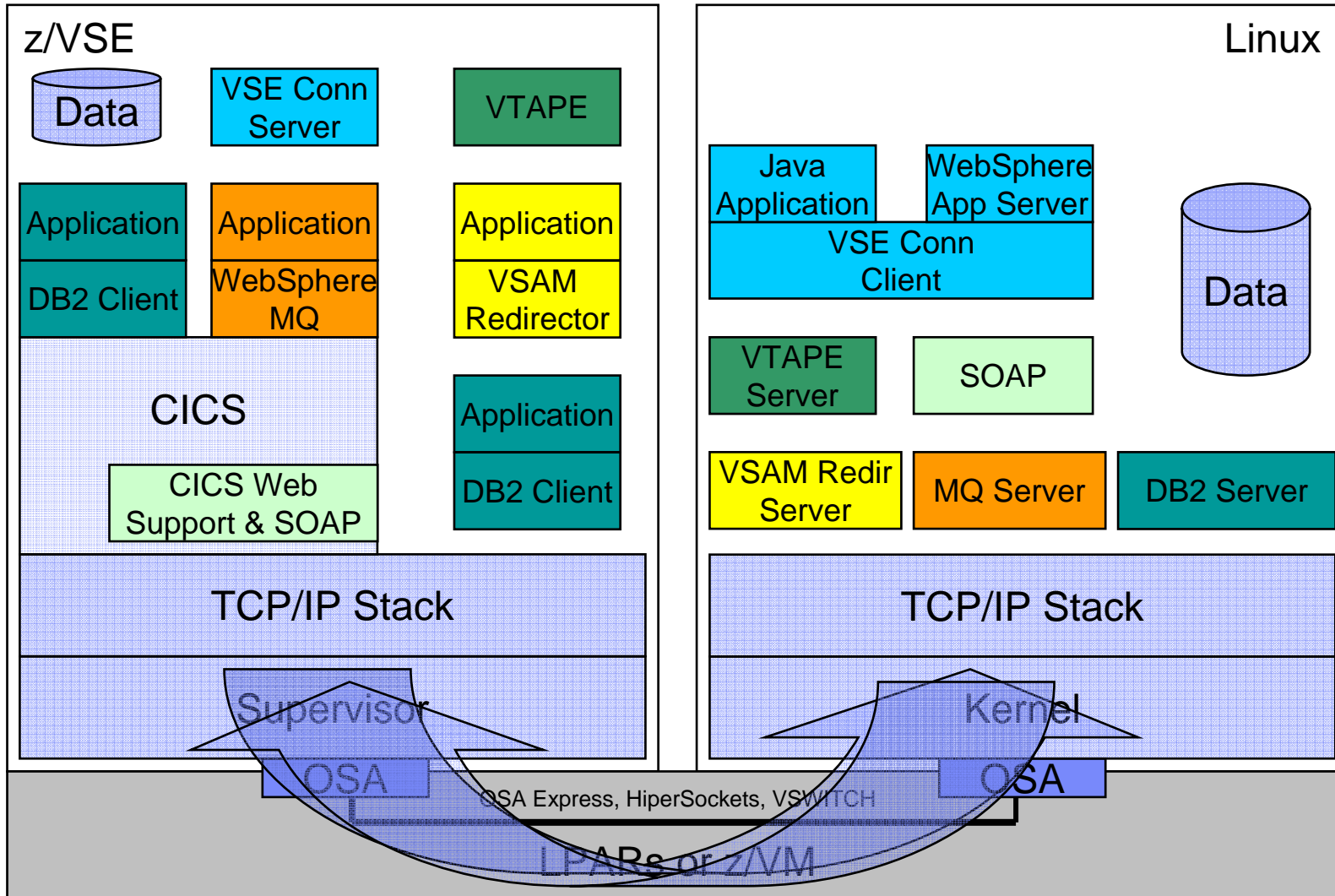


Agenda

- **Performance aspects for z/VSE applications with Linux in System z**
 - Typical problems & pitfalls
 - Tuning considerations
 - Monitoring Tools
 - VSAM versus Database

- **Parallel Access Volumes (PAV) Support in z/VSE**
 - Overview
 - Performance results

The big picture



Typical performance bottlenecks in such an environment

Throughput not as expected

Often seen with:

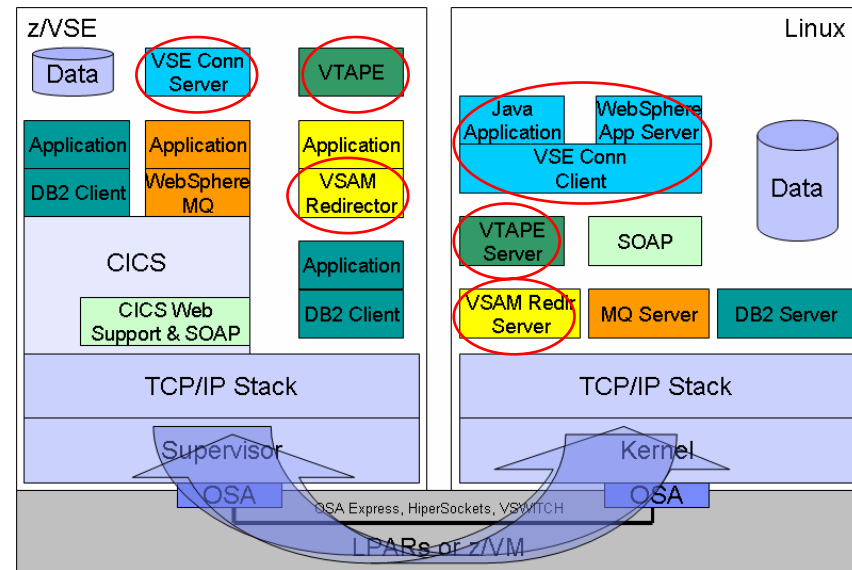
- File transfers (FTP)
- VTAPE
- VSE Connector Server & Client
- VSAM Redirector

Measurement methods:

- MB per second
- Records per second
- Duration for x MB or x Records

Limiting factors:

- CPU resources
- TCP/IP configuration
(e.g MTU size, Window Sizes)



Possible causes:

- Not enough CPU available
- Too much concurrent workload
- Incorrect Priority settings
- Unfavorable TCP/IP settings
- Too much network hops in-between
(Firewalls, VSWITCH, etc.)

Typical performance bottlenecks in such an environment

Response Time too high

Often seen with:

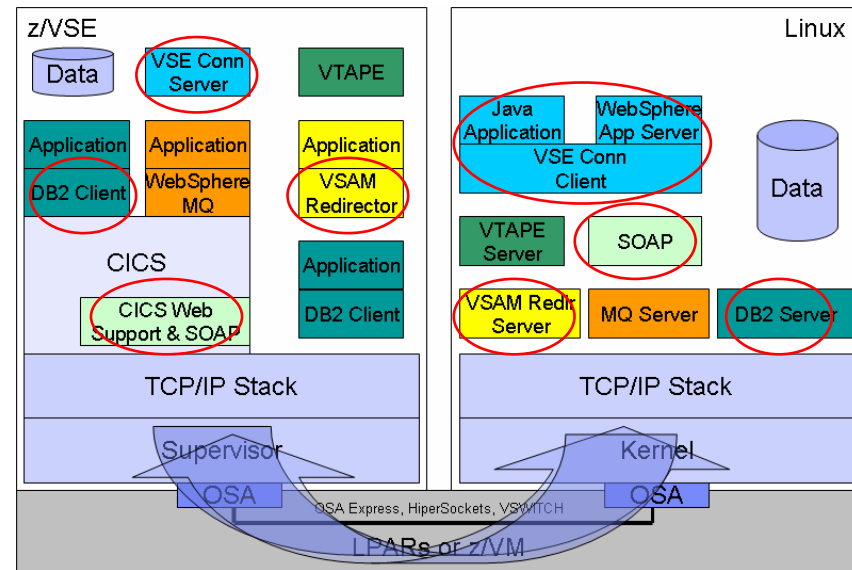
- DB2 Client/Server
- CICS Web Support & SOAP
- VSE Connector Server & Client
- VSAM Redirector

Measurement methods:

- Time between Request and Response
- Transactions per second

Limiting factors:

- CPU resources
- Priorities
- Network latency



Possible causes:

- Not enough CPU available
- Too much concurrent workload
- Incorrect Priority settings
- Unfavorable TCP/IP settings
- Too much network hops in-between (Firewalls, VSWITCH, etc.)

Typical performance bottlenecks in such an environment

CPU consumption too high

Often seen with:

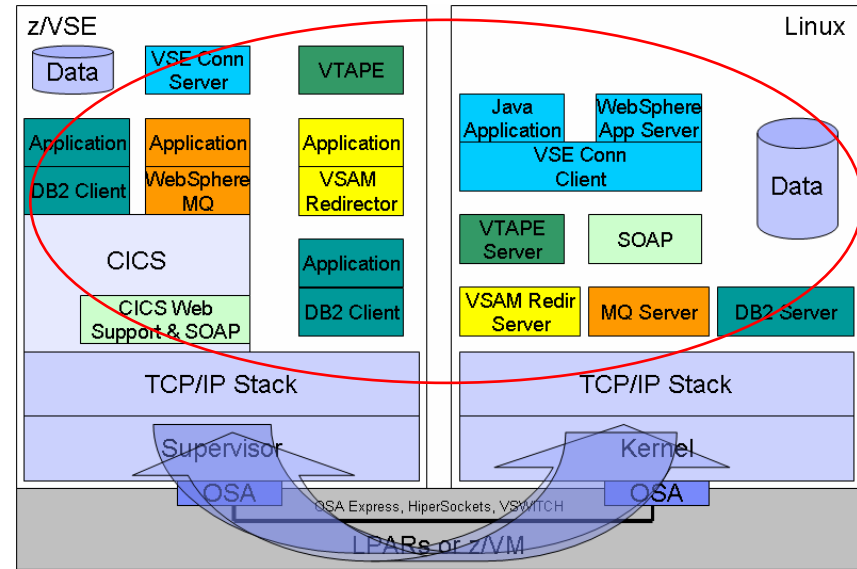
- All cases

Measurement methods:

- CPU utilization
- CPU time per job or transaction

Limiting factors:

- Available CPU resources
- Size of the box



Possible causes:

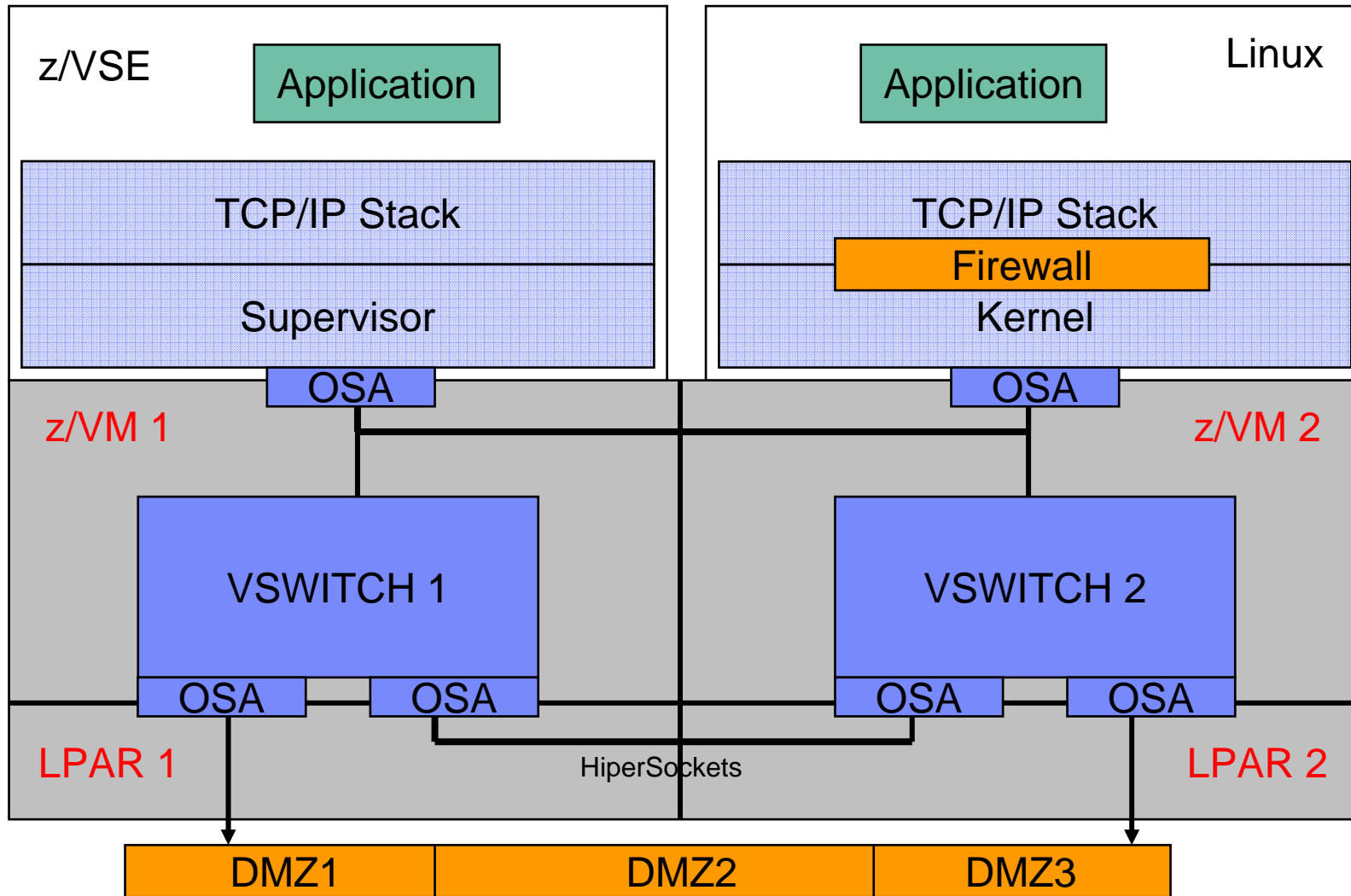
- Unfavorable TCP/IP settings
- Frequent network errors (retransmissions, misrouted packets, etc.)
- Applications not optimized for remote type of work

General rules for good performance

- **Higher throughput requires more CPU power**
 - The more packets to be delivered to the other side, the more CPU is required
 - But: More concurrent TCP/IP workload makes the stack behave more efficient
- **Better send data in less larger junks than in more smaller chunks**
 - Saves overhead per request
 - Buffer data in memory before sending it
- **Transfer only that much data that is required by the other side**
 - Don't transfer the whole file if only a single record is needed
- **Connection establishment is expensive**
 - Establish a connection once and keep it open for subsequent requests
 - Use connection pools wherever possible
 - Save unnecessary handshake (may include sign-on processing)
- **Reduce the number of network hops in-between client and server**
 - Remove unnecessary switches and routers
 - Firewalls may have a big impact on network performance
- **Consider Codepage translation overhead for textual data**
 - z/VSE is EBCDIC, Linux is ASCII



A simple picture might not be that simple in reality



Shared OSA Adapter versus HiperSockets

To connect a z/VSE system with a Linux on System z you have 2 options:

1. Using a shared OSA Adapter

- All traffic is passed through the OSA Adapter
- The OSA Adapter has **its own processor**
 - Processing occurs asynchronous
 - Processing in OSA Adapter does not affect host processors

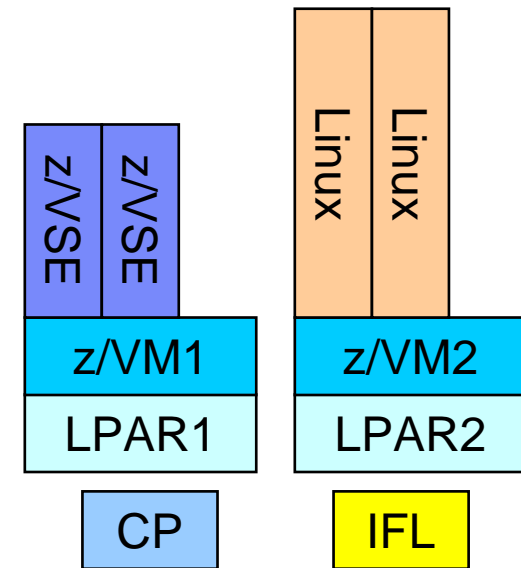
2. Using HiperSockets

- Direct memory copy from one LPAR/Guest to the other
- Memory copy is **handled by the host processors**
 - Processing occur synchronous
 - Consider mixed speed processors (full speed IFLs and throttled CPs)
 - Memory copy performed by throttled CP is slower than memory copy performed by full speed IFL



Considerations when sizing a system with z/VSE and Linux


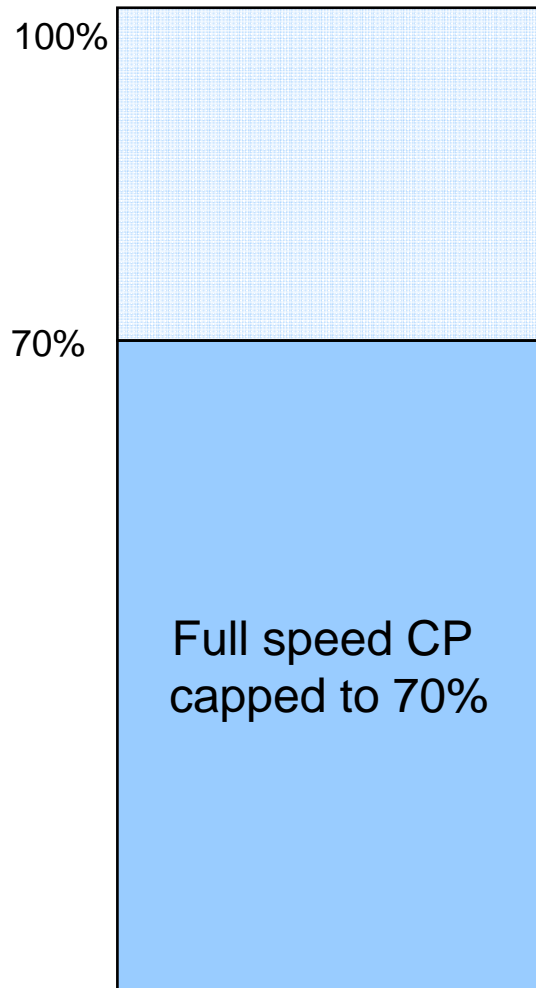
- Typically a such system has
 - 1 or multiple (full speed) IFLs
 - running one or multiple Linux images
 - 1 or multiple throttled CPs
 - running one or multiple z/VSE images
- Mostly Linux on System z runs in a Guest under z/VM
 - 1 or multiple images
- z/VSE may or may not run in a Guest under z/VM
 - 1 or multiple images
- The speed of the IFL processors is fix
 - dependent on the processor type (e.g. z9 BC, z10 BC, ...)
- The speed of a CP is dependent on the choosen capacity setting
- During the **Proof of Concept (PoC)** phase, the required size and number of the CPs and IFLs can be determined
 - It is always recommended to perform a proof of concept !
 - Have performance measurement tools in place on both sides



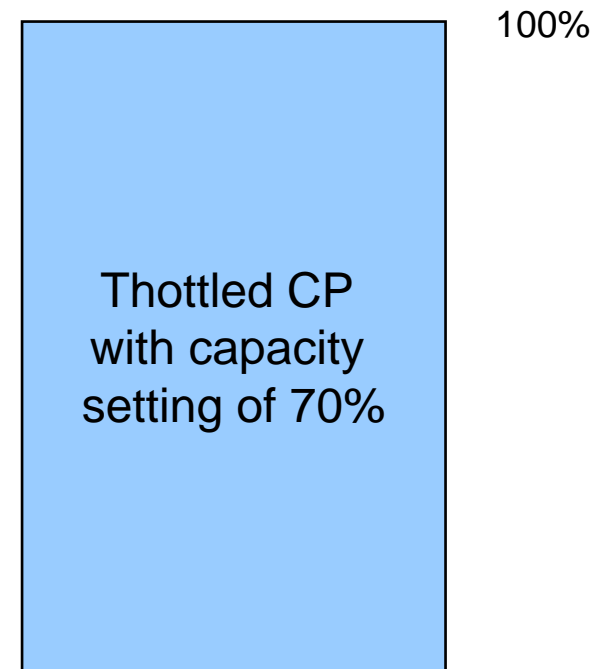
Capping versus Capacity Settings

Attention: Do not use Capping to simulate Capacity Settings !

- With **Capping**, the processor runs on its full speed, until the capping stops the guest from getting dispatched by the LPAR hipervisor or z/VM (timeslicing)
- With a **Capacity Setting**, the processor runs on a slower speed (and all related tasks as well, like Hipersockets memory copy, Hipervisor processing, etc)



Capping is NOT equivalent to Capacity Settings !

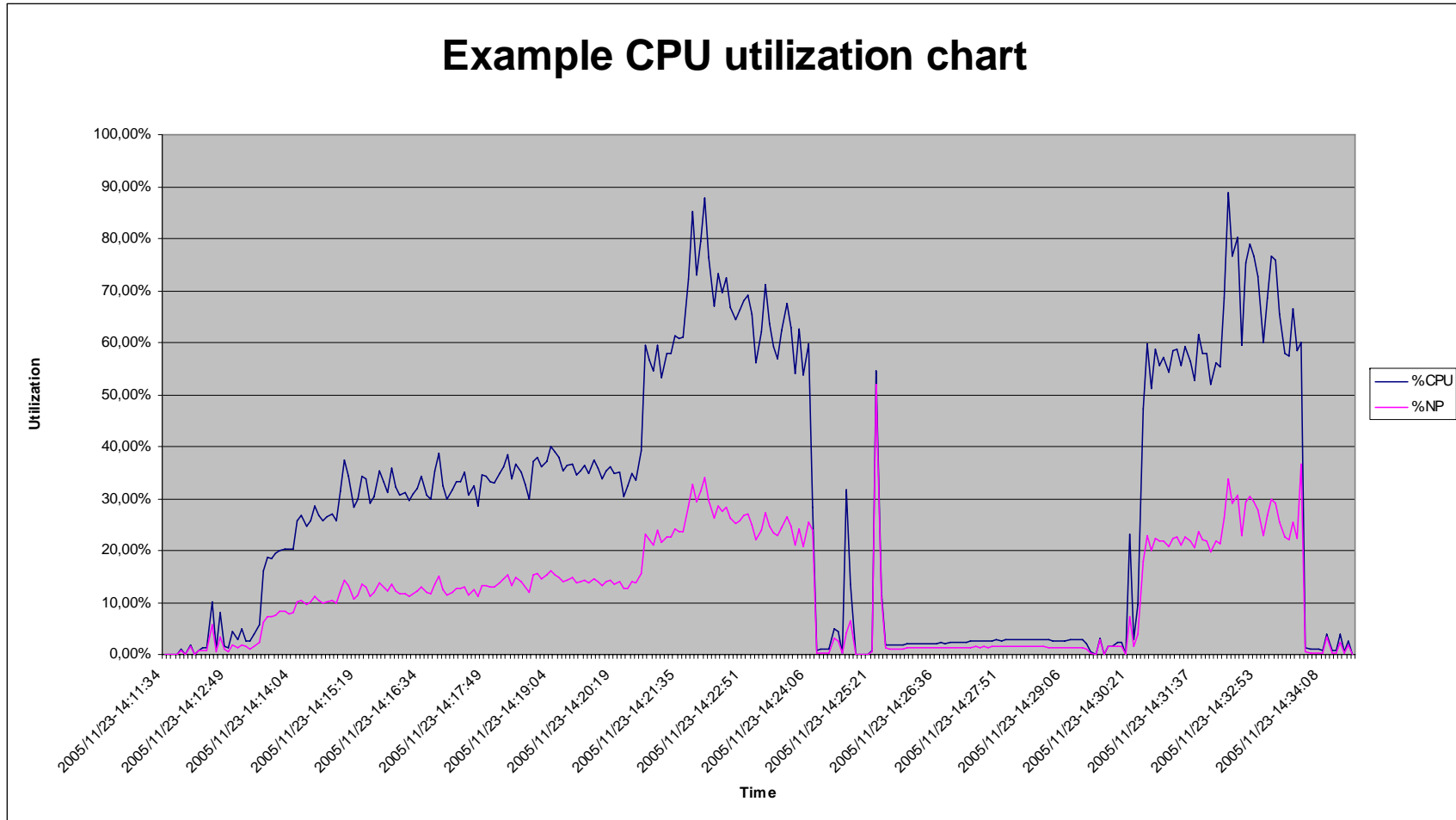


VSE CPU Monitor Tool

- Intended to help customers to **measure the CPU utilization** of their VSE system **over a period of time**.
- When you plan for a processor upgrade it is very important to know the CPU utilization of your VSE system over a day or a week.
 - Helps you to estimate the size of the new processor.
- The VSE CPU Monitor Tool is not intended to replace any existing monitoring product provided by partners.
- It provides only very **basic monitoring** capabilities on **an overall VSE system level**.
- No details about CPU usage of certain applications are provided
- New version available (XML Output) for z/VSE Capacity Planning
- **Download**
 - <http://www.ibm.com/systems/z/os/zvse/downloads/tools.html>
 - ‘As is’, no official support, e-mail to zvse@de.ibm.com



VSE CPU Monitor Tool



z/VSE Capacity Planning Offering

- A new **z/VSE Capacity Planning** Offering is now available
 - Available for Business Partners
 - and Customers
- Performance data collection is based on a new version of the CPUMON Tool
- Analysis is done using zCP3000
- Contact techline@us.ibm.com and ask for z/VSE Capacity Planning Support



Best practice activities for monitoring and tuning

Any performance improvement process includes the following fundamental steps:

1. Define the performance objectives.
2. Establish performance indicators for the major constraints in the system.
3. Develop and execute a performance monitoring plan.
4. Continually analyze monitoring results to determine which resources require tuning.
5. Make one adjustment at a time.

If an application or a set of applications does not run as fast as expected, you must become familiar with the affected systems and components.

- A distributed environment, such as z/VSE connected to a DB2 database server running on Linux on System z, requires a [look at the entire picture](#) rather than at each site or component separately.

The data and requests flow through several different software layers and services on its way from the z/VSE application to the server on Linux on System z and back.

- Each layer adds a certain amount of overhead and processing time.
- To learn why such an application does not perform as expected, you might need to [look at each layer](#), which includes the z/VSE application itself, the middleware running on z/VSE, the TCP/IP stack on both systems, the network, and middleware running on Linux on System z.
- In addition to the layers that are directly involved, you also must look at both operating systems, z/VSE and Linux on System z.



Best practice activities for monitoring and tuning

- **Remember the law of diminishing returns**
 - The greatest performance benefits usually come from your initial efforts.
- **Do not tune just for the sake of tuning**
 - Tune to relieve identified constraints.
 - If you tune resources that are not the primary cause of performance problems, you can make subsequent tuning work more difficult.
- **Consider the whole system**
 - You cannot tune one parameter or resource in isolation.
 - Before you make an adjustment, consider how the change will affect the system as a whole.
 - Performance tuning requires trade-offs among various system resources.
 - For example, you might increase buffer pool sizes to achieve improved I/O performance, but larger buffer pools require more memory, and that might degrade other aspects of performance.
- **Change one parameter at a time**
 - Do not change more than one factor at a time.
 - Even if you are sure that all the changes will be beneficial, you will have no way to assess the contribution of each change.



Best practice activities for monitoring and tuning

▪ Measure and configure by levels

- Tune one of the following levels of your system at a time:
 - Hardware
 - Operating system
 - Middleware components
 - Application programs



▪ Check for both hardware and software problems

- Some performance problems can be corrected by applying service to your hardware, your software, or both.
- Do not spend excessive time monitoring and tuning your system before applying service to the hardware or software.

▪ Understand the problem before you upgrade your hardware

- Even if it seems as though additional storage or processor power might immediately improve performance, take the time to understand where the bottlenecks are.
- You might spend money on additional disk storage, only to find that you do not have the processing power or the channels to exploit it.

▪ Put fallback procedures in place before you start tuning

- If tuning efforts result in unexpected performance degradation, reverse the changes that are made before you attempt an alternative approach. Save your original settings so that you can easily undo changes that you do not want to keep.

Best practice activities for monitoring and tuning

1. Collect data:

- The data collection step involves the usage of various performance monitor tools and optionally tracing tools on all components that are affected
- Part of this step proves that the monitoring and tracing tools work as expected and produce the kind of output that is needed for further analysis.

2. Analyze the data:

- The data analysis step uses the data that was collected in the previous step and analyzes it.
- Depending on the performance problem, concentrate on unusual behavior that catches your attention such as delays, high CPU utilization, long wait times, high water marks, and so on.

3. Tune the systems:

- Based on the results from analyzing the data, you tune the different parts of the environment.
- The goal is to change settings that influence the behavior when analyzing the data but in a positive way.

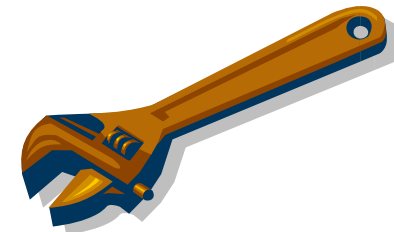
4. Run the test case:

- Re-run the test case to check whether the tuning has helped.
- Don't forget to activate the performance monitor and tracing tools
- Make sure that the preconditions of the test case are the same on all test runs



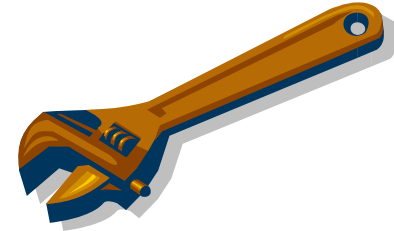
z/VSE monitoring tools

- **System Activity Dialogs** (SYSA fast path 361 and 362)
 - Displays real-time performance information about the System, CPU, partitions and I/O
- **QUERY TD** command
 - Displays information about CPU usage on the console
- **SIR SMF** command
 - Displays I/O related performance information on the console
- **Job Accounting Exit** (SKJOBACC in ICCF library 59)
 - Prints performance related information (CPU, I/O) to SYSLST after each job step
- **MAP** and **GETVIS** commands
 - Displays memory related information on the console
- **z/VSE CPUMON Tool**
 - Monitors overall system CPU usage and performance counters
- **CICS Statistics**
 - Prints CICS statistics
- **CICS built-in tools** like CEMT INQUIRE
 - Displays information about CICS resources
- A **z/VSE performance monitor** product for batch and CICS
 - Like CA Explore, ASG TMON, etc.

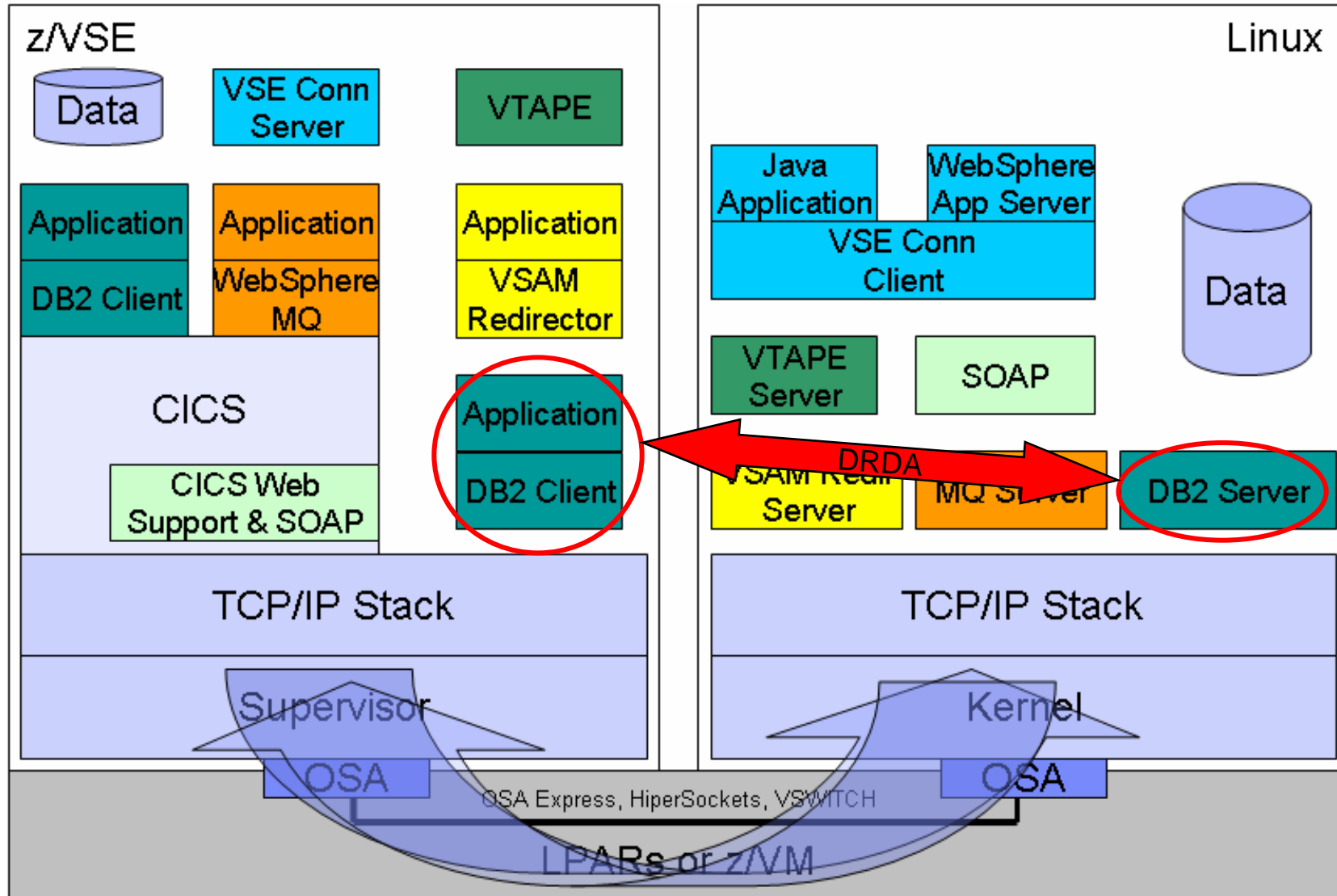


Linux and z/VM monitoring tools

- **sysstat** utilities
 - A collection of performance monitoring tools for Linux
- **iostat** utility
 - Monitors disk utilization
- **top** utility
 - Prints system data for each process. It shows an overview of the currently running system processes.
- **oprofile** utility
 - profiles all running code on Linux systems, providing a variety of statistics
- **IBM Tivoli OMEGAMON XE** on z/VM and Linux
 - Displays data that is collected from multiple systems in one, flexible interface called the Tivoli Enterprise Portal
- **Performance Toolkit for z/VM**
 - Provides capabilities to monitor and report performance data for a z/VM system
- **3rd Party Performance Monitoring Tools**
 - e.g. Tools from Velocity Software, Inc

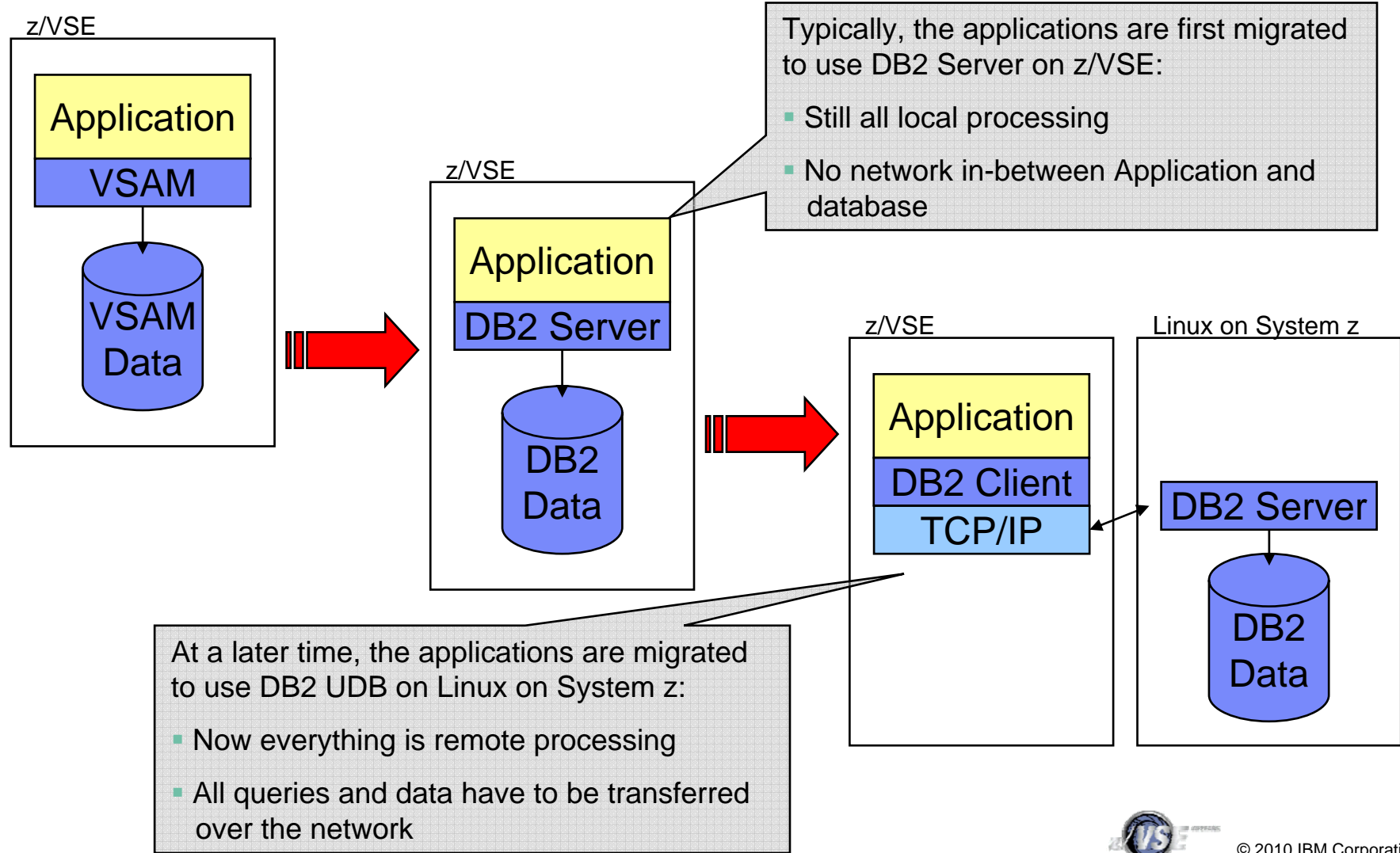


Lets have a closer look at one typical scenario



Best practices when migrating a VSAM application to DB2

Typical migration path from VSAM to DB2:



Best practices when migrating a VSAM application to DB2

Fundamental differences between VSAM and a database

VSAM

- It is an **access method**, not a database
- Optimized for **sequential** processing
- Good performance due to **read-ahead** in buffers in memory
 - Good chance that next record to be read is already in memory
- Works with **complete records**
 - No field level information is known by VSAM
 - Only the KEY portion of the record is known by VSAM
- One request affects just one record at a time
- Records have a certain order (e.g. by key)
- Processing occurs all locally inside z/VSE

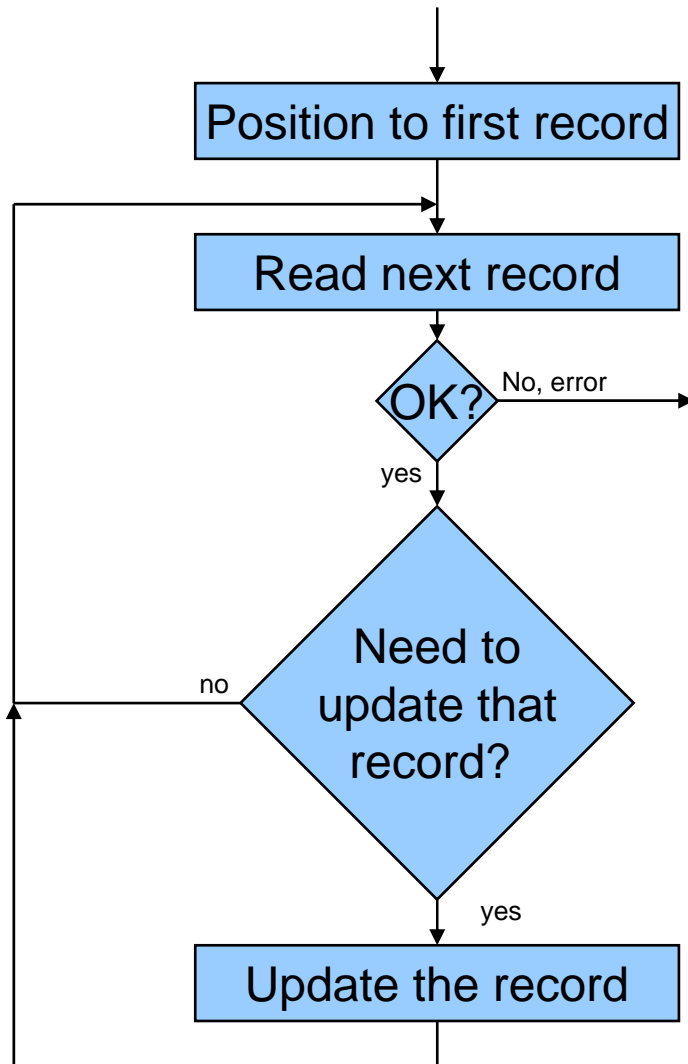
Database

- Is usually a **multi-user system**
- Optimized for **relational** processing
- Good performance due to **relational** constraints
- Data selection can be based on (complex) searches (WHERE clauses)
 - Every statement has its own individual access plan
- One statement can affect multiple rows, fields and even tables at once
 - Example: UPDATE ... WHERE ...
- Rows in a table does not have an order
 - The query can use an ORDER BY clause to request the data to be returned in a certain order
- Processing typically occurs remotely



Best practices when migrating a VSAM application to DB2

Example for a typical VSAM application processing flow:



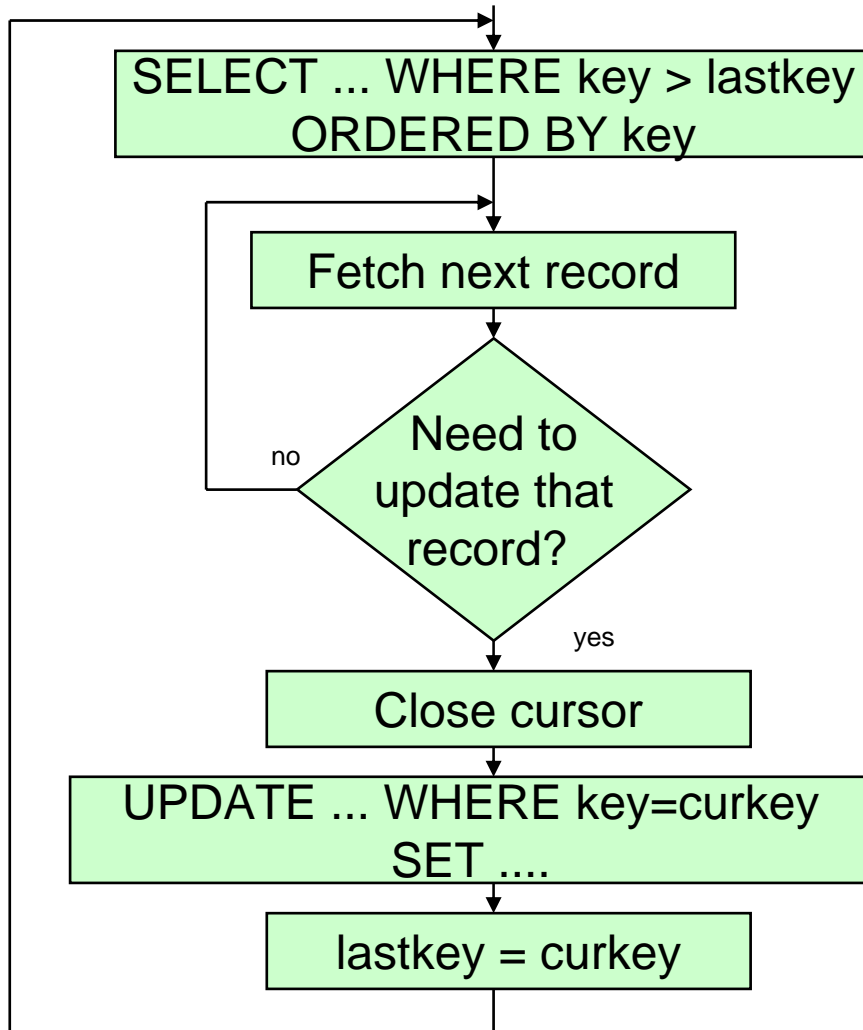
- Simple & straight forward processing logic
- Based on **sequential access**
- Decision whether the record needs to be updated is done **in the programs logic**
- Reading records that are not be updated does not cause much overhead (VSAM read-ahead)

Performs good !



Best practices when migrating a VSAM application to DB2

Same processing but now with a database (simple/stupid approach):



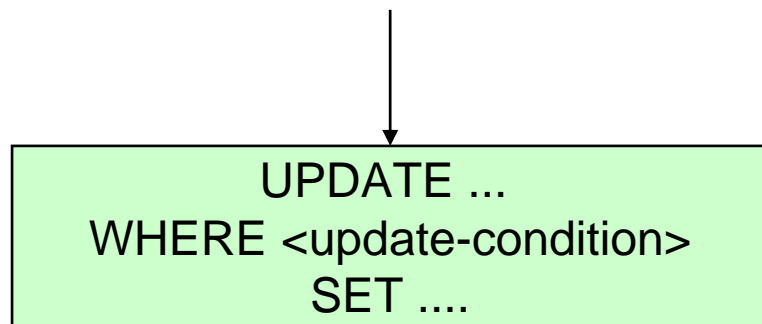
- Simple straight forward processing logic
- Still based on **sequential access** (!)
- Decision whether the record needs to be updated is still done **in the programs logic** (!)
- Reading records that are not be updated cost **a lot of overhead**
 - Network transfer of result set
 - SELECT statements executed several times
 - SORT processing for open ended WHERE

Performs bad !



Best practices when migrating a VSAM application to DB2

Same processing now optimized for a relational database:



Performs good !

- Pretty simple SQL statement
- Based on **relational database access**
- Decision whether the record needs to be updated is now done **by the database**
 - Optimized in the database using indexes, etc.
- **No more reading** of records that are not be updated
- **No data transfer** required between application and database (just the statement itself)
- But: The application logic has to be changed !

General rules of thumb for best performance:

- **If you change the data from sequential to relational, also change the application from sequential processing to relational processing**
- **Think relational** if you are working with a database, **forget VSAM**



Best practices for applications using DB2 Client

▪ DB2 CONNECT considerations

- Use [connection pooling](#) whenever possible
 - [Logical connections](#) are established by means of DB2 CONNECT command
 - [Physical connections](#) are realized by establishing a TCP/IP connection to the database server
- Establishing physical connections can be time consuming
 - Nameserver lookup
 - TCP/IP connection establishment
 - Handshaking



▪ Authentication considerations

- During DB2 CONNECT processing the database server performs [authentication](#) to the underlying operating system
 - Some CICS transactions perform a CONNECT statement every time a new program is invoked
 - Results in high numbers of CONNECT statements being executed
- With SUSE Linux Enterprise Server 9 (SLES 9) the default [password encryption algorithm](#) has changed from DES to BLOWFISH
 - The BLOWFISH algorithm can not be accelerated with hardware crypto (CPACF)
 - May cause a huge amount of overhead and increased CONNECT duration

Best practices for applications using DB2 Client

SQL Query tuning

→ **Explain every SQL statement !**

Explain tells you:

- The sequence of operations performed to process the query (access plan)
- Cost information
- Statistic for all objects referenced by the statement
- Information about indexes used by the query



Possible actions:

- Create additional indexes
- Reorganize indexes
- Simplify the SQL queries
- Add processing hints to the query

Best practices for applications using DB2 Client

Application design guidelines for best performance



- **Let the database determine which rows to update**
 - Through a WHERE clause
 - Do not sequentially read all rows to perform the update decision in your application code

- **SELECT statements should return small result sets**
 - Query only those rows that your application really needs

- **Do not use SELECT statements with open ended WHERE clauses**
 - Example: “SELECT WHERE column >= value ORDERD BY column”
 - Result set contains all rows from value up to the end of the table !
 - Very expensive if SORT is involved

- **If you can not limit the result set by using a proper WHERE clause, use**
 - OPTIMIZE FOR n ROWS clause
 - FETCH FIRST n ROWS ONLY clause

Best practices for applications using DB2 Client

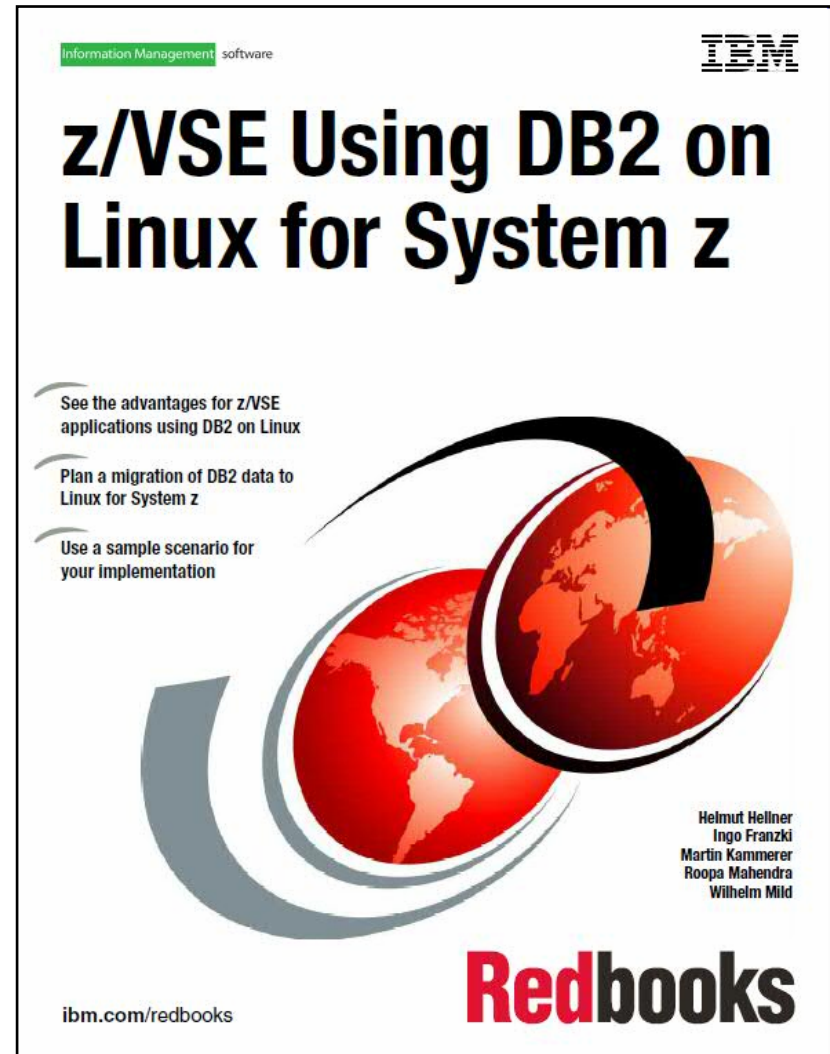
Application design guidelines for best performance (continued)

- **Transfer as little data as possible with every statement or result set**
 - If no further rows are required from a result set, stop fetching and close the result set
 - This does not save on overhead for creating the huge result set, but at least it saves on overhead for transferring the remaining rows
 - Only include those columns into a SELECT statement that you really need
 - Do not use „SELECT * FROM table“
- **Result set data is transferred in larger chunks**
 - Check DB2 BLOCK SIZE
- **Avoid unnecessary SORTs whenever possible**
 - If the order of the result rows is not important, do not use the ORDER BY clause.



New RedBook: z/VSE Using DB2 on Linux for System z

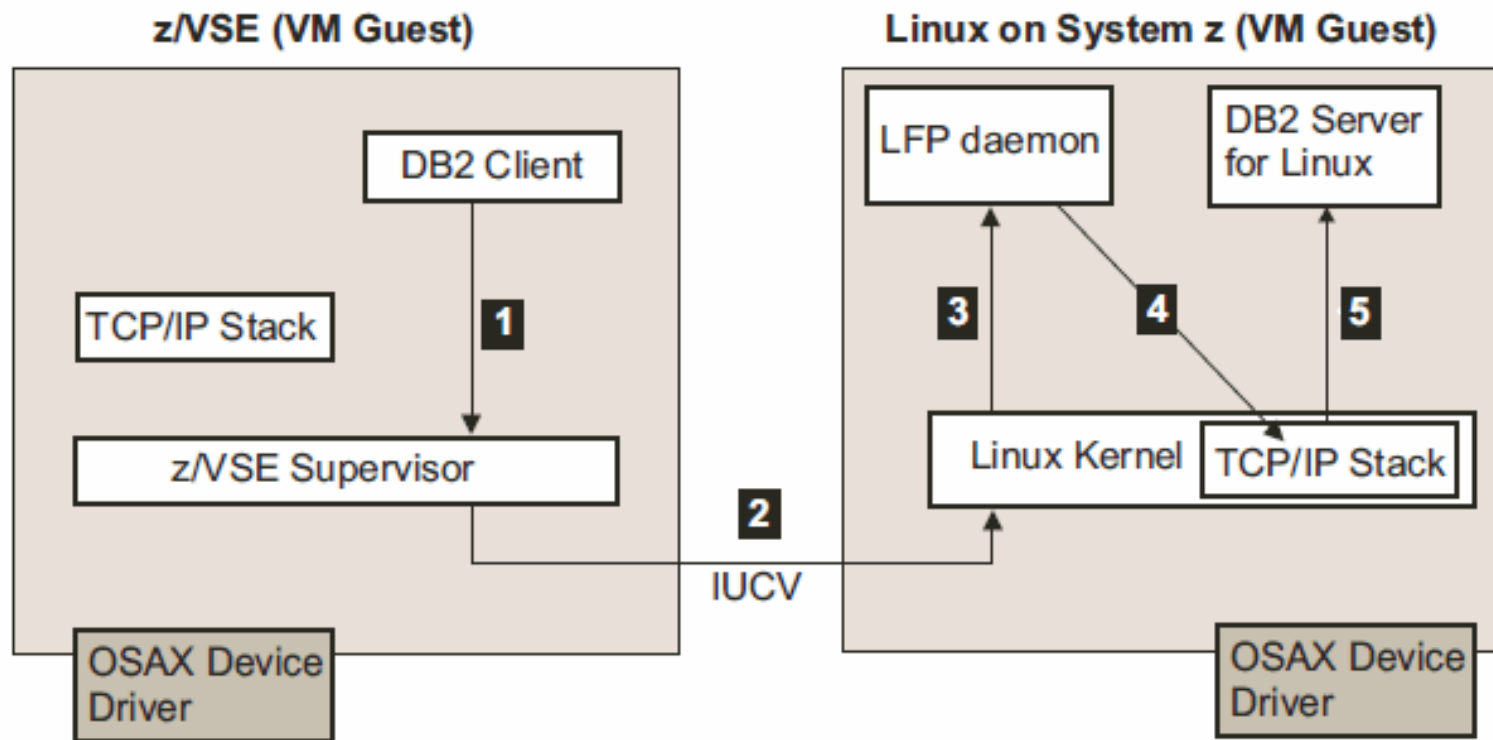
- Available since February 03, 2010
- IBM Form Number: SG24-7690
- <http://www.redbooks.ibm.com/redpieces/abstracts/sg247690.html?Open>
- Contents:
 - Chapter 1. Overview of a future oriented DB2 environment
 - Chapter 2. Planning DB2
 - Chapter 3. Environment setup and customization
 - Chapter 4. DB2 data migration and application dependencies
 - Chapter 5. Monitoring and tuning
 - Appendix A. Configuration members
 - Appendix B. Database manipulation



New: z/VSE 4.3 – Fast Path to Linux on System z



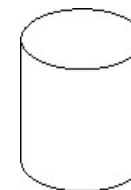
- The Linux Fast Path uses an IUCV connection between z/VSE and Linux
 - Both systems run in the same **z/VM-mode LPAR** on a System z10 or z196
- It allows selected TCP/IP applications to communicate with the TCP/IP stack on Linux **without using a TCP/IP stack on z/VSE.**



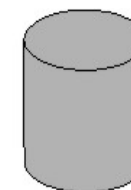
z/VSE V4.2 – Parallel Access Volumes (PAV)



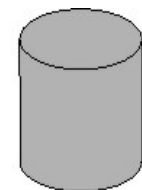
- Parallel Access Volume (PAV) is an **optional licensed feature** on the IBM System Storage DS6000/DS8000 series
- Parallel access volumes:
 - Are managed by creating **multiple addresses (aliases)** for a single **logical device**.
 - Allow your z/VSE system to **access volumes in parallel**.
 - Enable **more than one I/O operation** to be processed for a single logical device.
 - Depending upon workload, might represent a significant performance improvement by the storage unit over traditional I/O processing.
 - Can significantly **reduce device queue delays** to busy devices (all I/O operations are normally processed from a single queue).
 - Allow your system to access a single volume from a single z/VSE host with **multiple concurrent requests**.



Base



Alias X



Alias Y

z/VSE V4.2 – Parallel Access Volumes (PAV)

- To gain a performance benefit from PAV, it is required that:
 - the workload can **utilize multiple paths** (sub channels) to the same volume
 - A single job that **starts an I/O and waits** until its completion before it starts another I/O **does not benefit** from PAV by itself
 - However, **running multiple such jobs concurrently** in multiple partitions can utilize parallel access to the same volume.

- Besides the simple ‘Start I/O and wait’ behavior, more sophisticated programs (like CICS) may be capable to **start multiple I/Os and wait for its completion asynchronously**
 - This behavior also is eligible for benefiting from PAV, if the I/Os are for the same volume

- Jobs that use multiple subtasks that perform I/O operations may benefit from PAV

- Running **multiple jobs concurrently** in separate partitions and access the same volume are also good candidates for PAV



z/VSE V4.2 – Parallel Access Volumes (PAV)

- Candidates for PAV volumes
 - In general, all volumes that contain files, catalogs of data in general that are **used from multiple jobs** can benefit from PAV
 - Without PAV, I/Os might get queued and therefore delayed in case of parallel access to the same volume
 - PAV allows parallel access to these files from different jobs without disturbing each other.
 - VSAM Catalogs
 - Shared VSAM Clusters
 - VSE Libraries

- Besides volumes that contain files that are shared between multiple jobs, also volumes that contain **heavily used system files** can benefit from PAV
 - Although there might not be any parallel access to those system files as such, there may be **parallel access to other files residing on the same volume**
 - Without PAV, I/Os might get queued and therefore delayed in case of parallel access to the same volume
 - PAV allows parallel access to these volumes without being disturbed by access to the system files.
 - POWER Spool files
 - Workfiles
 - Log files



z/VSE V4.2 – Parallel Access Volumes (PAV)

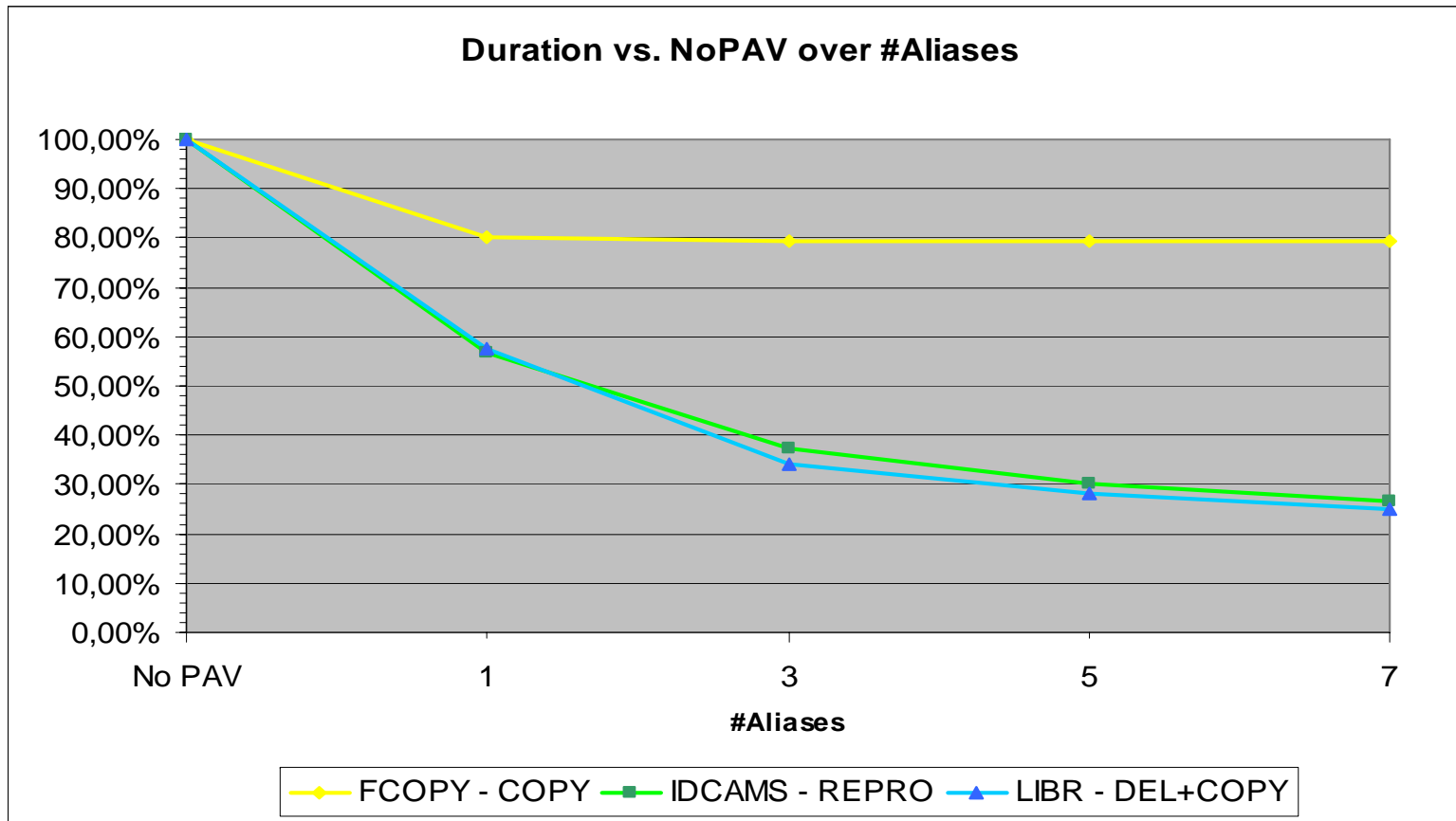
- Besides the general PAV setup, **no changes** in application programs or JCL are required
 - PAV is enabled in z/VSE via the SYSDEF SYSTEM command:

```
SYSD E F  S Y S T E M  
//      , P A V =  S T A R T  
                S T O P
```

- Define the PAV base devices to z/VSE via ADD statement
 - You must NOT define your PAV-alias volumes to z/VSE!
 - An ADD statement for an alias device will be ignored
- z/VM support for PAV (since z/VM 5.2)
 - Dedicated PAV devices
 - PAV Mindisks.
 - For more information see here:
<http://www.vm.ibm.com/storman/pav/pav2.html>

z/VSE V4.2 – Parallel Access Volumes (PAV)

PAV performance measurements:



Conclusion: Dependent on the workload, PAV can dramatically increase the I/O throughput and reduce the job duration (e.g. 1/3 of the duration when running without PAV), if it is not limited by other factors (like LOCKS).



Questions ?

