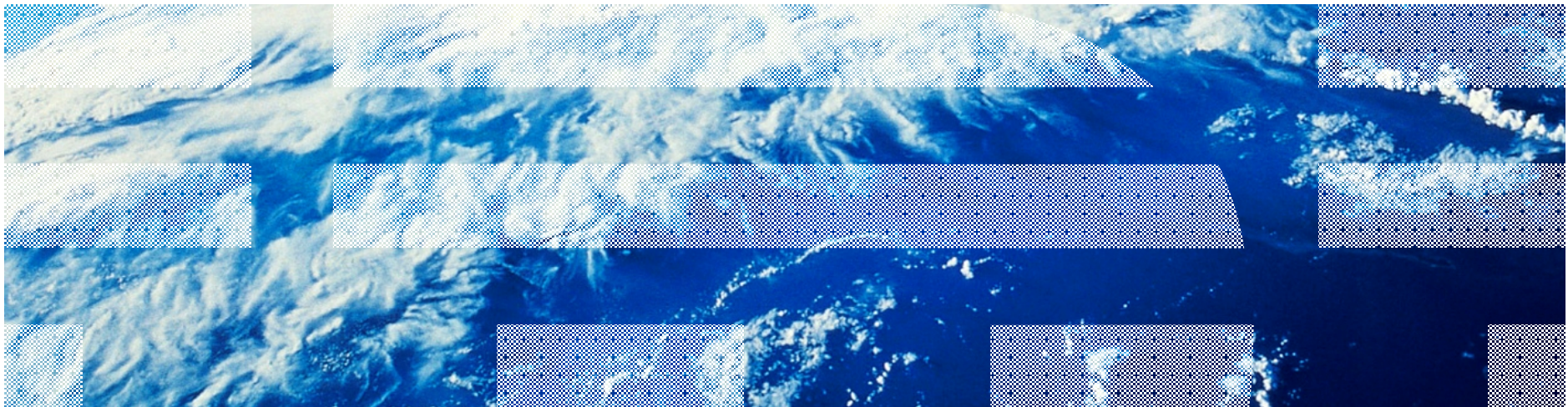# Performance Tuning of z/VSE Applications working with Linux on System z

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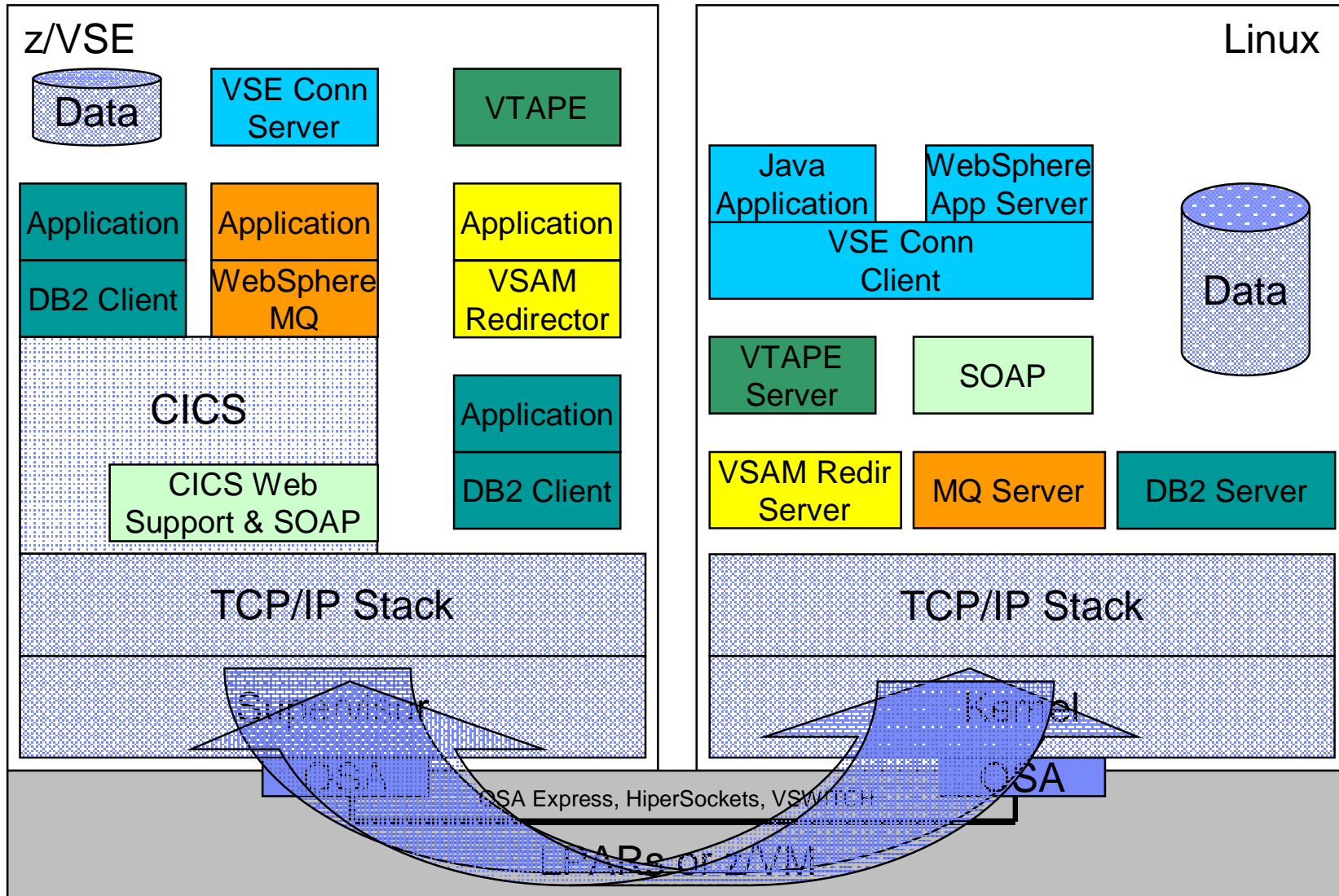This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available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

IBM

The big picture

z/VSE

Data

VSE Conn Server

VTAPE

Application

DB2 Client

Application

WebSphere MQ

Application

VSAM Redirector

CICS

CICS Web Support & SOAP

Application

DB2 Client

TCP/IP Stack

Supervisor

OSA

Linux

Java Application

WebSphere App Server

VSE Conn Client

Data

VTAPE Server

SOAP

VSAM Redir Server

MQ Server

DB2 Server

TCP/IP Stack

Kernel

OSA

OSA Express, HiperSockets, VSWITCH

LPARs or z/VM

© 2010 IBM Corporation

Typical performance bottlenecks in such an environment

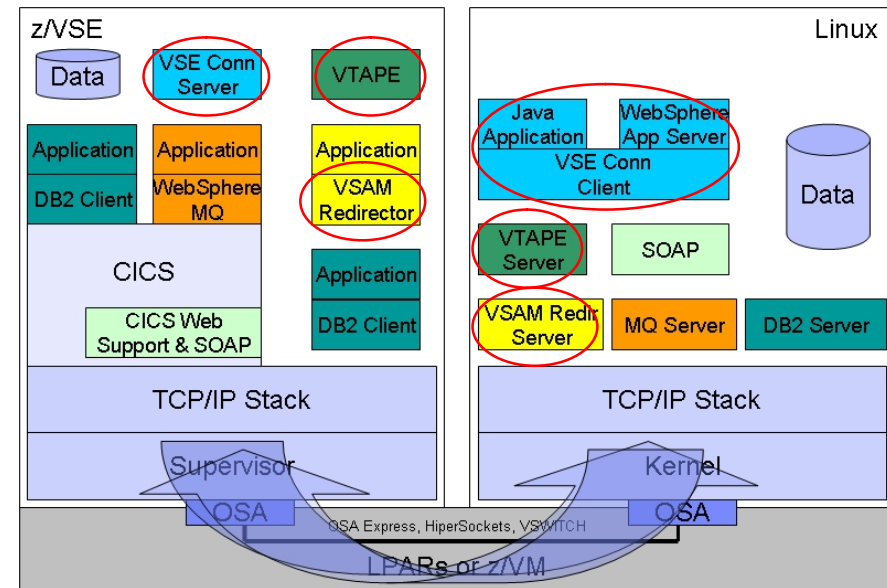## Throughput not as expected

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
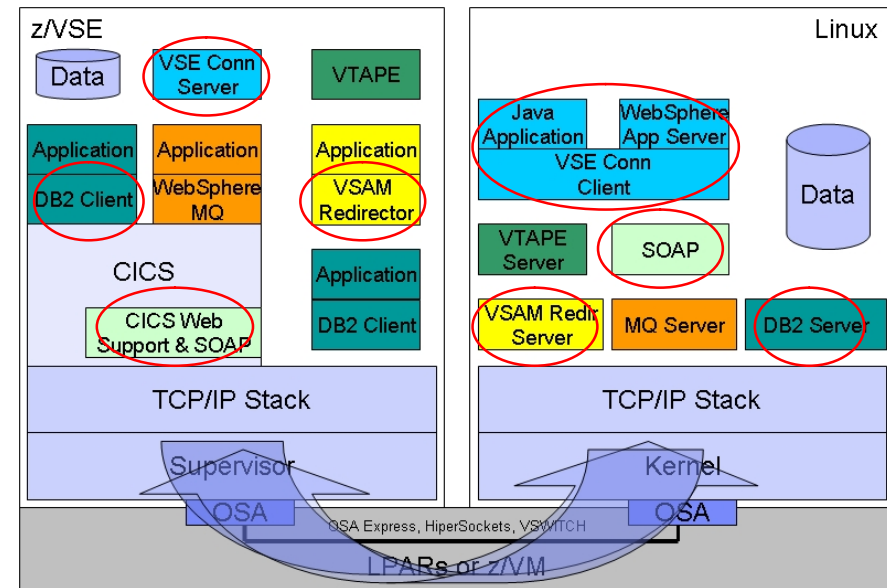
Response Time too high

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
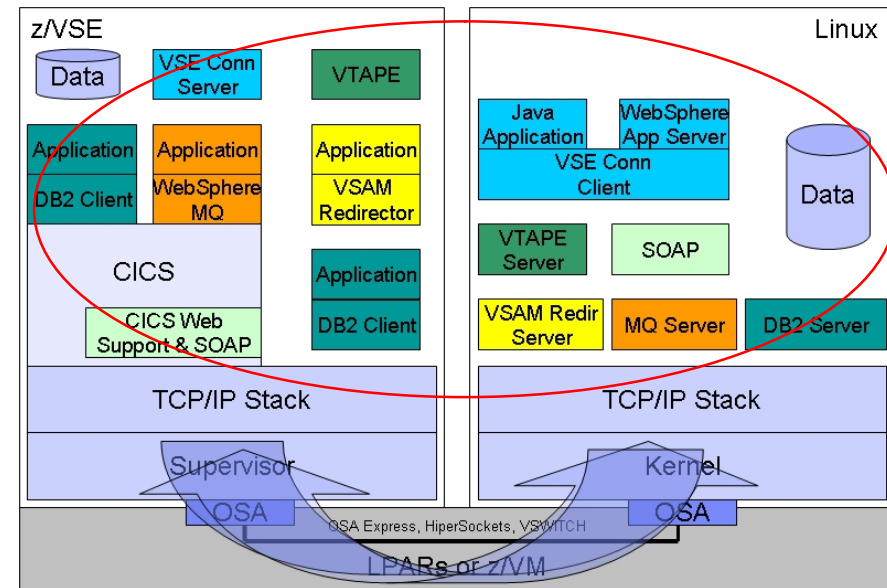
CPU consumption too high

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 possibe
- Save unnecessary handshake (may include sign-on processing)

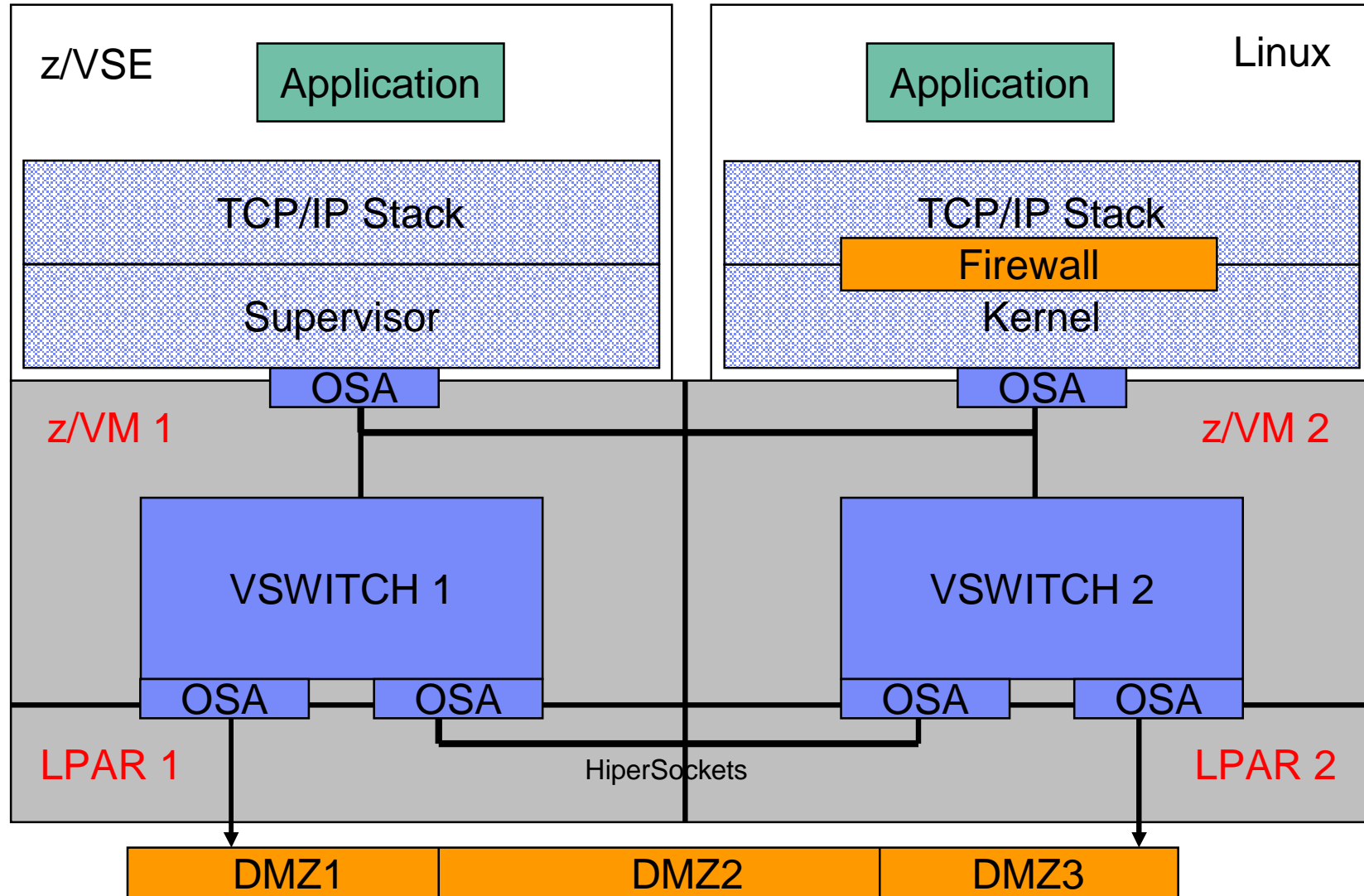§ **Reduce the number of network hops in-between client and server**
- Remove unnecessary switches and routers
- Firewalls may have a big impact on network performance

§ **Consider Codepage translation overhead for textual data**
- z/VSE is EBCDIC, Linux is ASCII

A simple picture might not be that simple in reality

z/VSE

Linux

Application

Application

TCP/IP Stack

Supervisor

TCP/IP Stack

Firewall

Kernel

OSA

OSA

z/VM 1

z/VM 2

VSWITCH 1

VSWITCH 2

OSA

OSA

OSA

OSA

LPAR 1

HiperSockets

LPAR 2

DMZ1

DMZ2

DMZ3

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)
      à 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 tp perform a proof of concept !
  – Have performance measurement tools in place on both sides

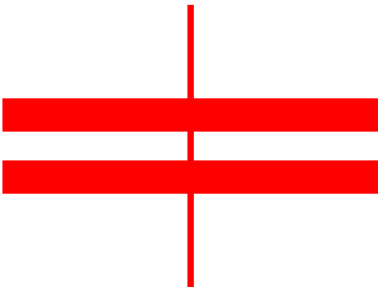| z/VSE	z/VSE		Linux	Linux
z/VM1			z/VM2	
LPAR1			LPAR2	
CP			IFL	
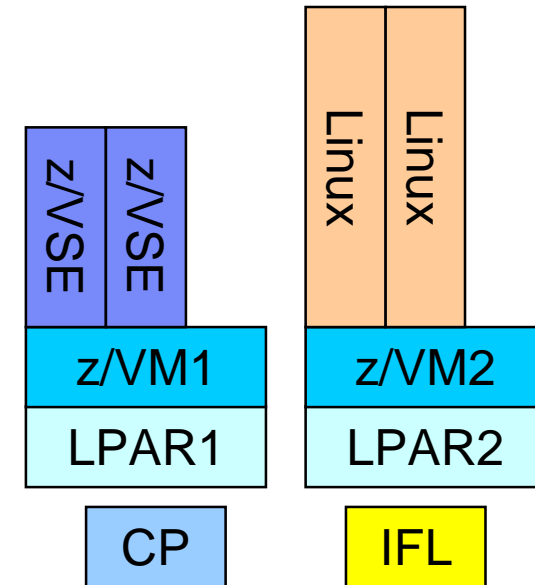
Capping versus Capacity Settings

Attention: Do <u>not</u> 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

§ With a **Capacity Setting**, the processor runs on a slower speed (and all related tasks as well, like HiperSockets memory copy, Hipervisor processing, etc)

100%

70%

100%

Full speed CP capped to 70%
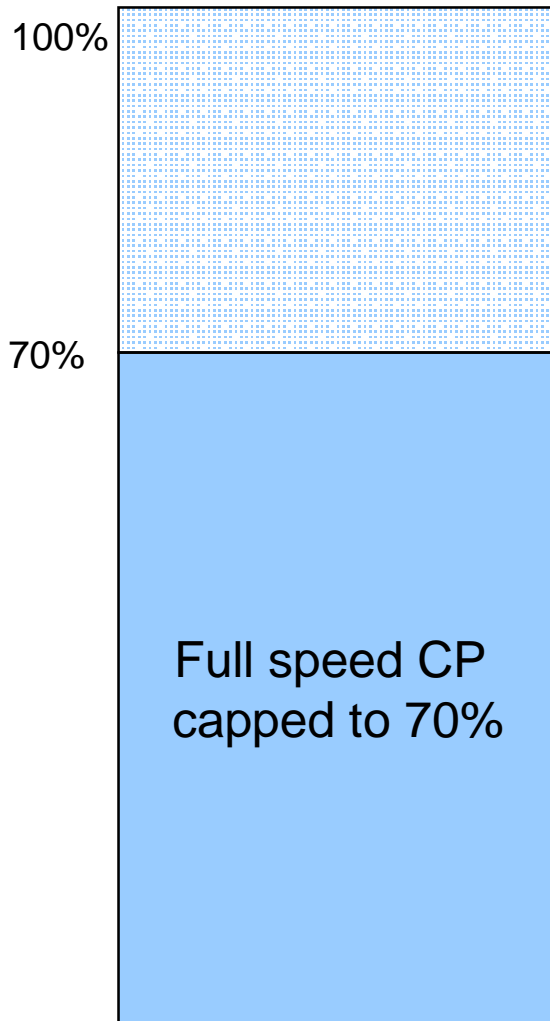
Capping is NOT equivialent to Capacity Settings !

Thottled CP with capacity setting of 70%
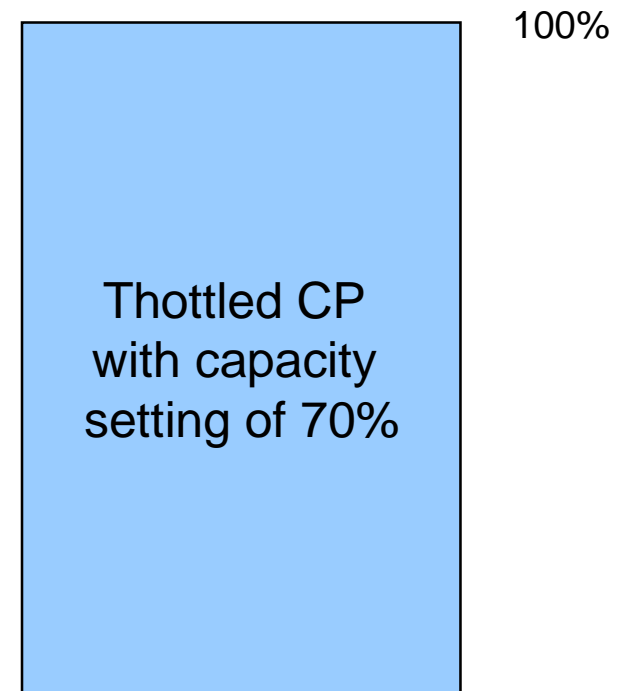
Best practice activities for monitoring and tuning

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 (SYS 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 ressources

§ A z/VSE performance monitor product for batch and CICS
– Like CA Explore, ASG TMON, etc.
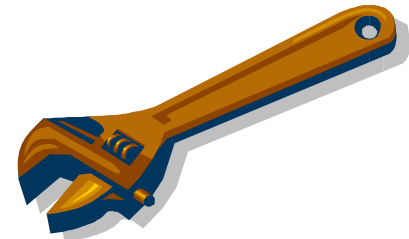
Linux and z/VM monitoring tools

§ 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
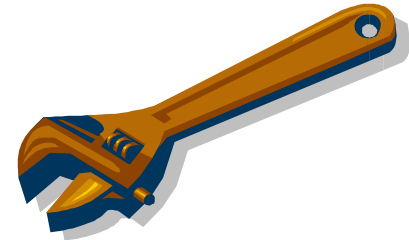
Lets have a closer look at one typical scenario

Best practices when migrating a VSAM application to DB2

Typical migration path from VSAM to DB2:

z/VSE

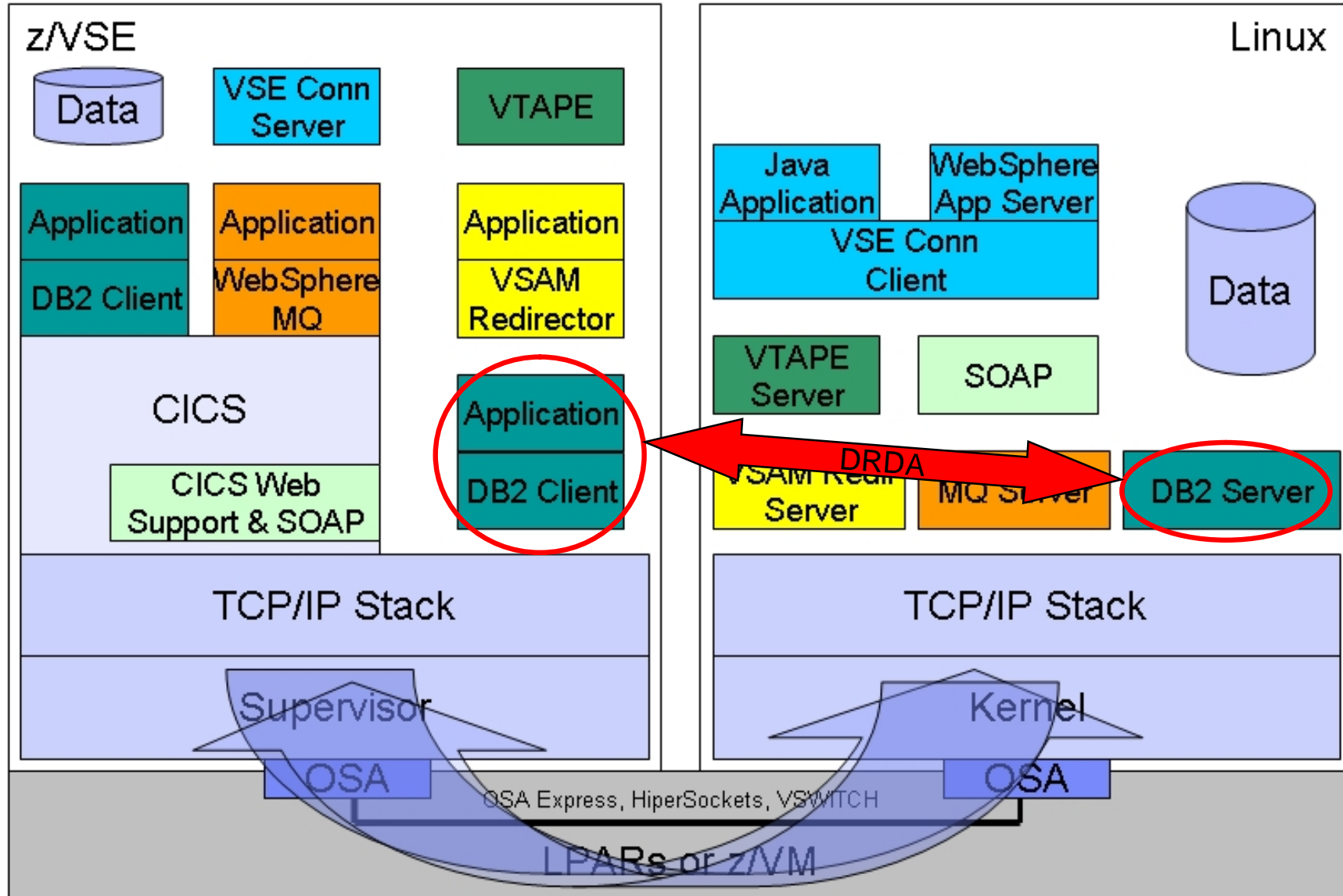| Application |
| VSAM |

VSAM Data

z/VSE

| Application |
| DB2 Server |

DB2 Data

Typically, the applications are first migrated to use DB2 Server on z/VSE:

§ Still all local processing

§ No network in-between Application and database

z/VSE

| Application |
| DB2 Client |
| TCP/IP |

Linux on System z

| DB2 Server |

DB2 Data

At a later time, the applications are migrated to use DB2 UDB on Linux on System z:

§ Now everything is remote processing

§ All queries and data have to be transferred over the network

19

© 2010 IBM Corporation

Best practices when migrating a VSAM application to DB2

Fundamental differences between VSAM and a database

VSAM
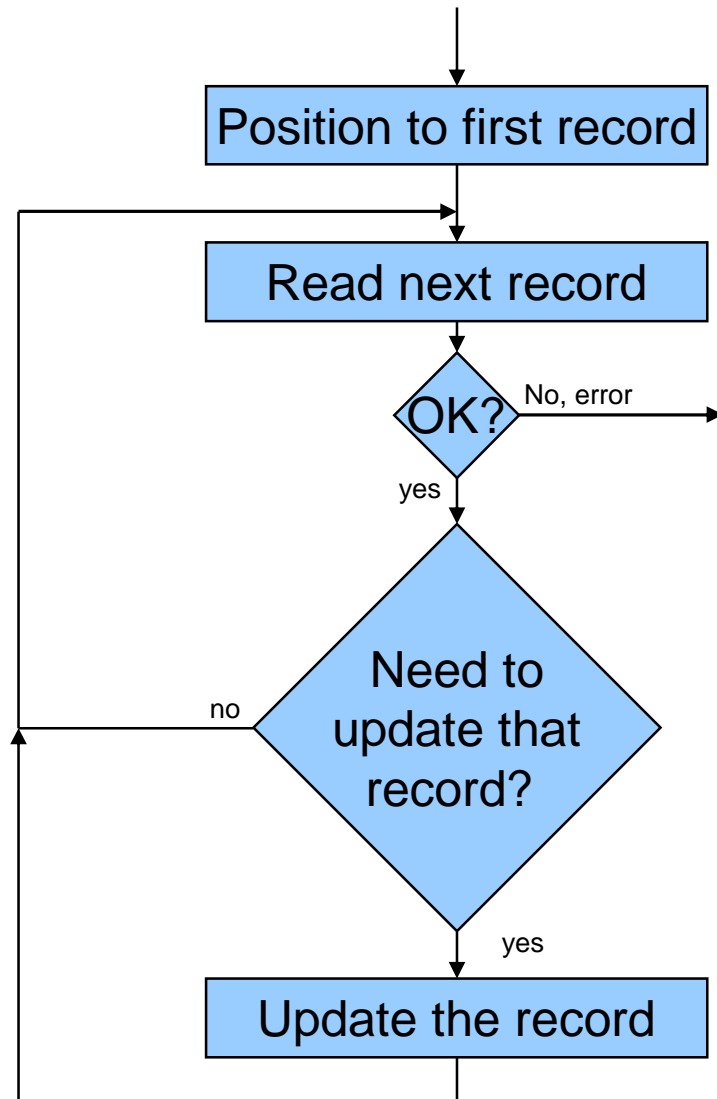
§ It is an access method, not a database

§ 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

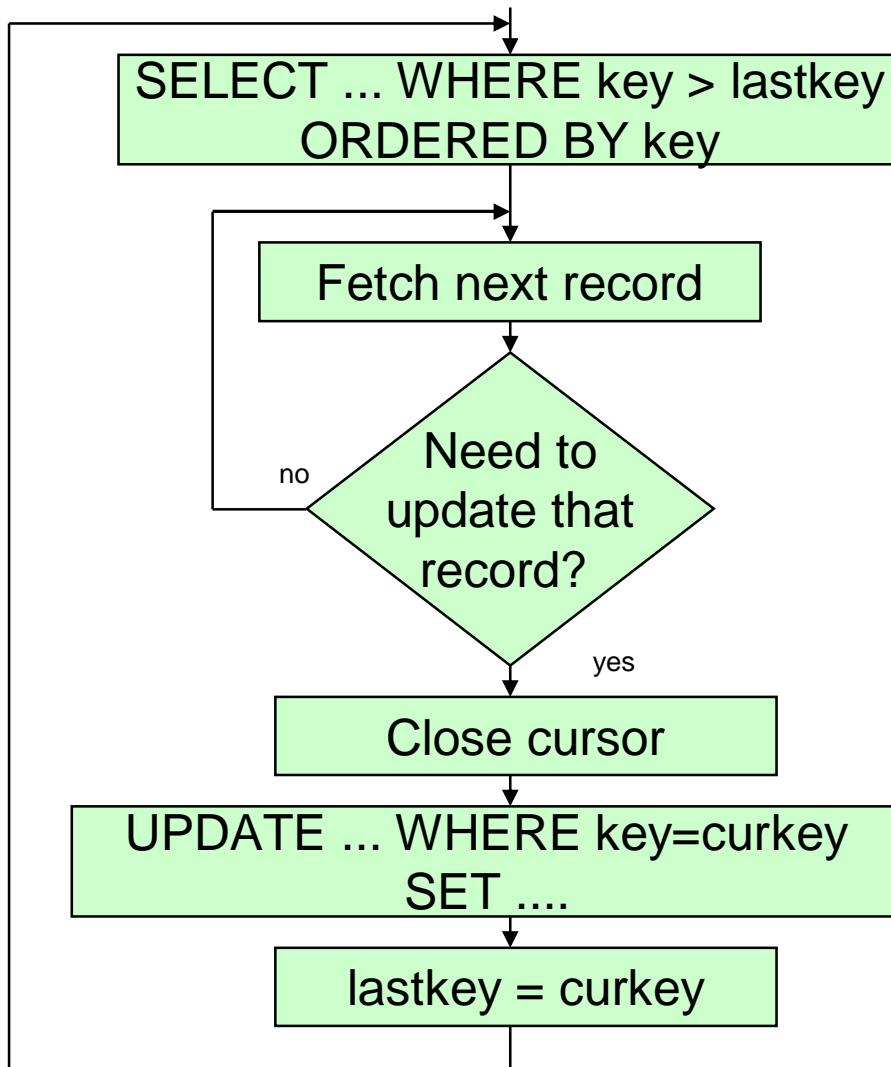Example for a typical VSAM application processing flow:

```
        |
        v
  ┌───────────────────────┐
  │ Position to first record │
  └───────────────────────┘
        |
        v
  ┌───────────────────────┐
  │   Read next record    │
  └───────────────────────┘
        |
        v
      < OK? >  ──No, error──>
        |
       yes
        |
        v
   < Need to
     update that     ──no──>
     record? >
        |
       yes
        |
        v
  ┌───────────────────────┐
  │   Update the record   │
  └───────────────────────┘
```

§ 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):


```
SELECT ... WHERE key > lastkey
ORDERED BY key
```

Fetch next record

Need to update that record?

no

yes

Close cursor

```
UPDATE ... WHERE key=curkey
SET ....
```

lastkey = curkey

§ 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
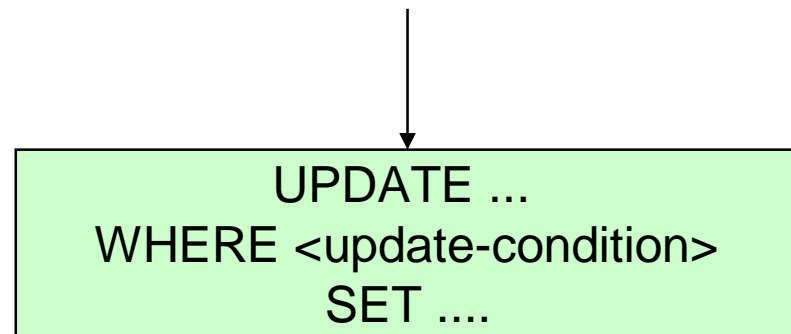
  – Network transfer of result set
  – SELECT statements executed several times
  – SORT processing for open ended WHERE

Performs bad !

© 2010 IBM Corporation

Best practices when migrating a VSAM application to DB2

Same processing now optimized for a relational database:

UPDATE ...
WHERE <update-condition>
SET ....

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:

à If you change the data from sequential to relational,
   also change the application from sequential processing to relational processing

à 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

à **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 <u>not</u> 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 <u>not</u> 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 ORDERD BY clause.
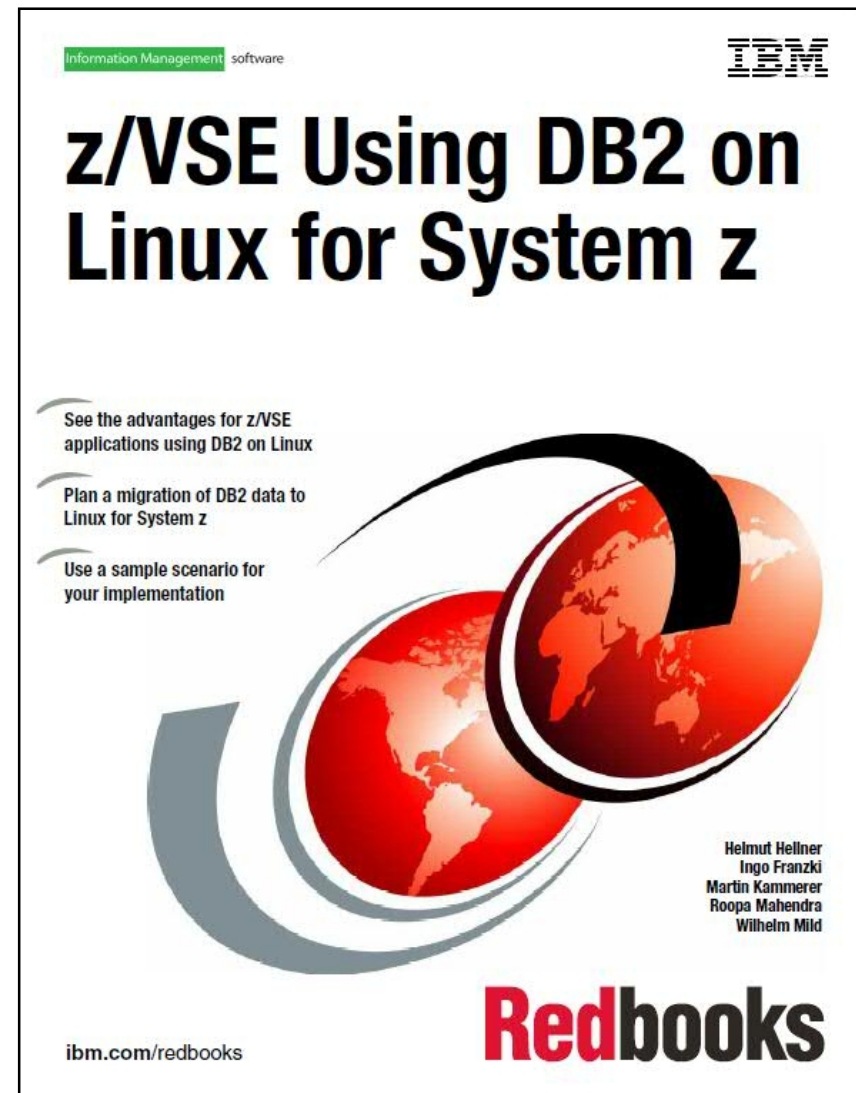
New RedBook: z/VSE Using DB2 on Linux for System z

§ 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

Information Management software

IBM

z/VSE Using DB2 on Linux for System z

See the advantages for z/VSE applications using DB2 on Linux

Plan a migration of DB2 data to Linux for System z

Use a sample scenario for your implementation

Helmut Hellner
Ingo Franzki
Martin Kammerer
Roopa Mahendra
Wilhelm Mild

Redbooks

ibm.com/redbooks

Summary

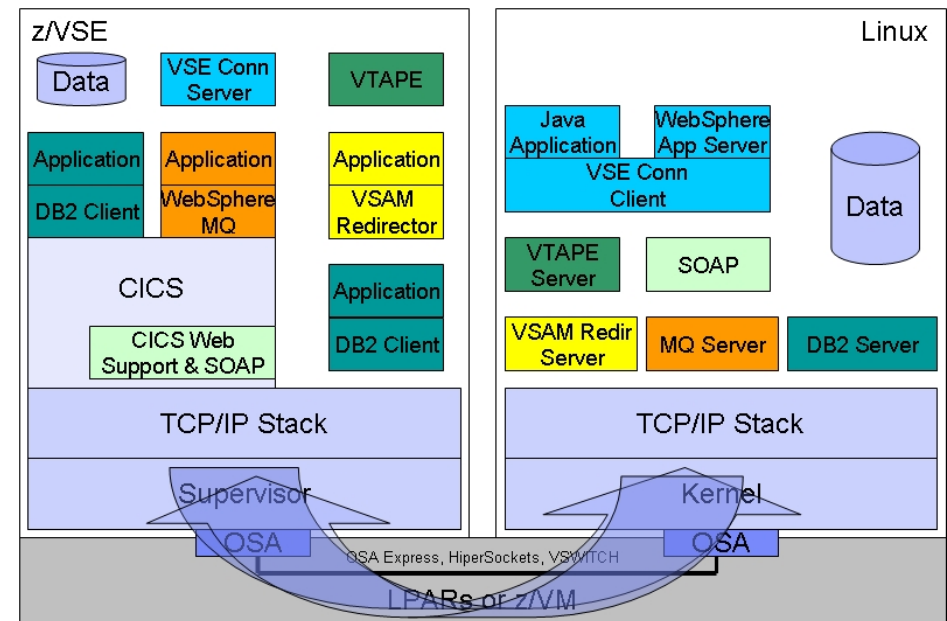§ Every scenario has different performance implications

§ Tuning may involve
  – Tuning of the environment
    • Hardware
    • Software
    • Network
  – Adaptions to the applications
    • Sometimes the biggest performance boost can be achieved by changing the application

§ Have proper monitoring tools in place

§ Multiple cycles are required of:
  – Collecte data
  – Analyse data
  – Tune the systems
  – Rerun the testcase

Questions ?