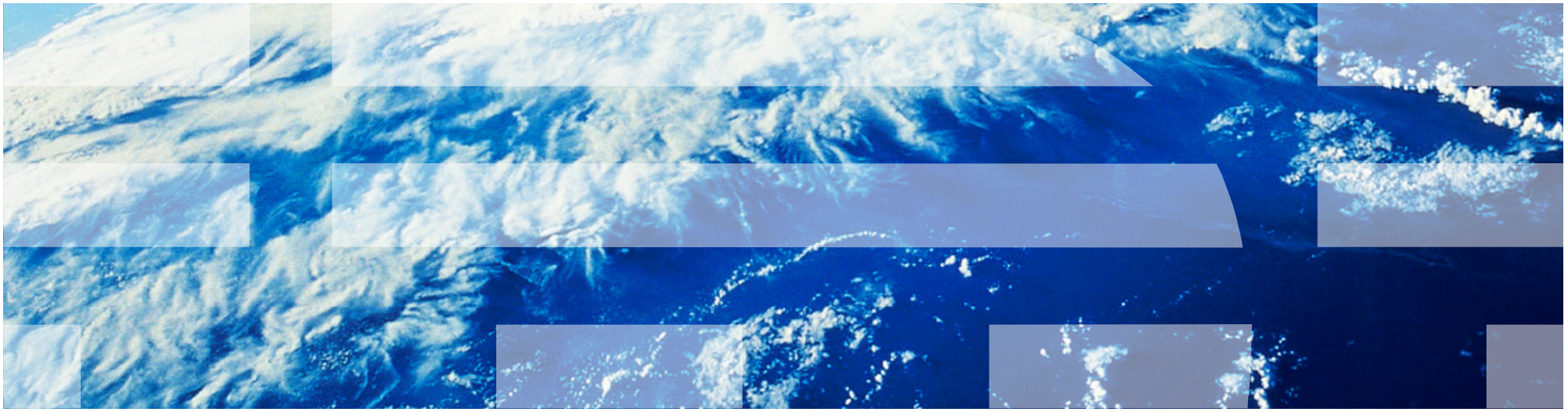
2013-12-09

Linux on System z - z/VM 6.2 Live Guest Relocation with Linux Middleware

Dr. Juergen Doelle
Michael Johanssen



IBM Research and Development
Boeblingen Germany

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

The following are trademarks or registered trademarks of other companies.

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- SUSE is a registered trademark of Novell, Inc. in the United States and other countries.
- Red Hat, Red Hat Enterprise Linux, the Shadowman logo and JBoss are registered trademarks of Red Hat, Inc.
  in the U.S. and other countries.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Agenda

1. **z/VM system considerations**
   - **z/VM Single System Image (SSI) cluster**
   - **z/VM Live guest relocation (LGR)**
     - ISFC logical link configuration
     - FICON CTC devices
   - **ISFC logical link configuration and relocation performance**
     - Number of FICON channels
     - Number of FICON FCTC devices
     - Configuration variations
     - Optimized setup
2. **Linux guest and application considerations**
   - **Comparing System z key middleware workloads**
     - Java™ application
     - Filesystem application
     - Transactional database application
3. **Resolving a resource constrained scenario with LGR**
4. **Summary**

PART I

z/VM system considerations

Basic structure of a z/VM SSI cluster

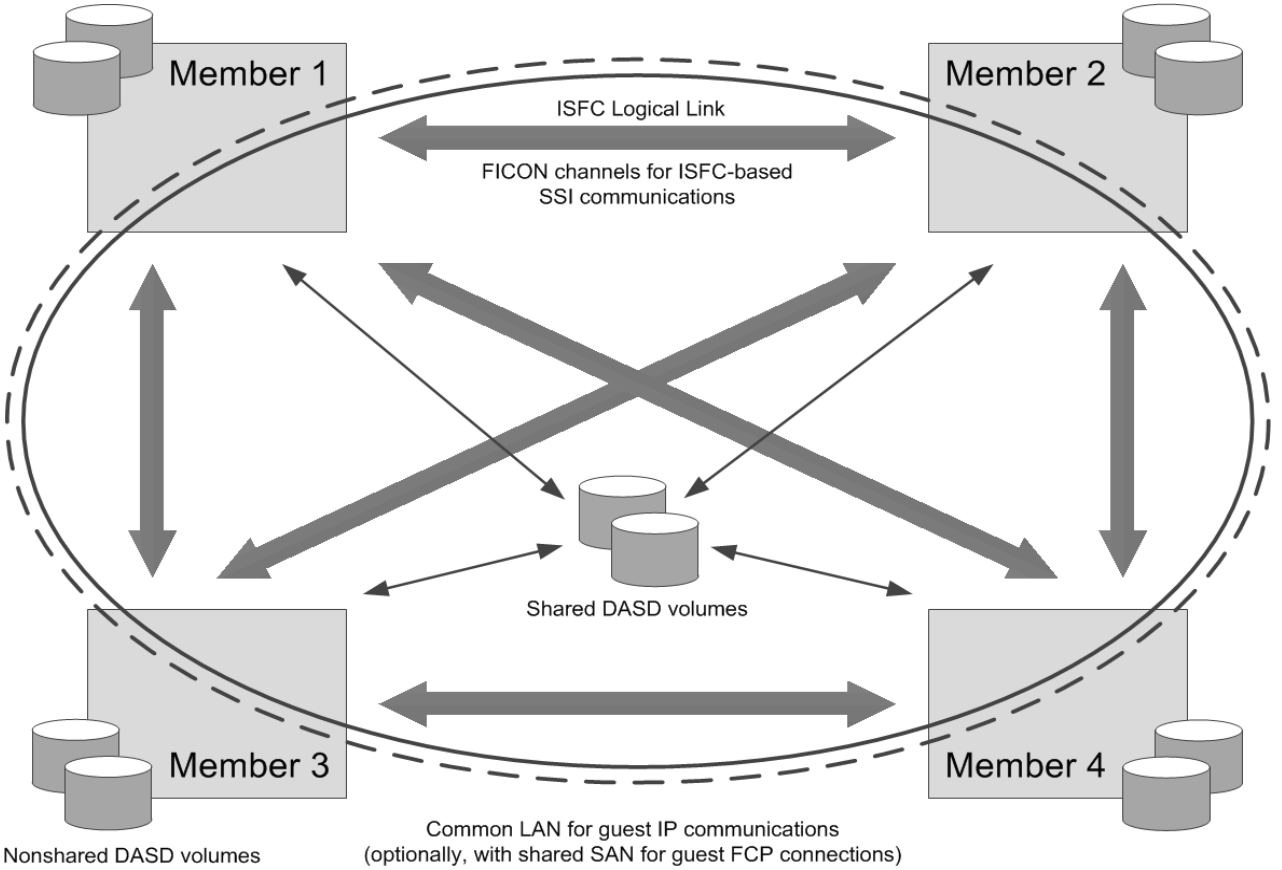
SSI
Single System Image

ISFC
Inter-System Facility for Communications

LGR
Live Guest Relocation

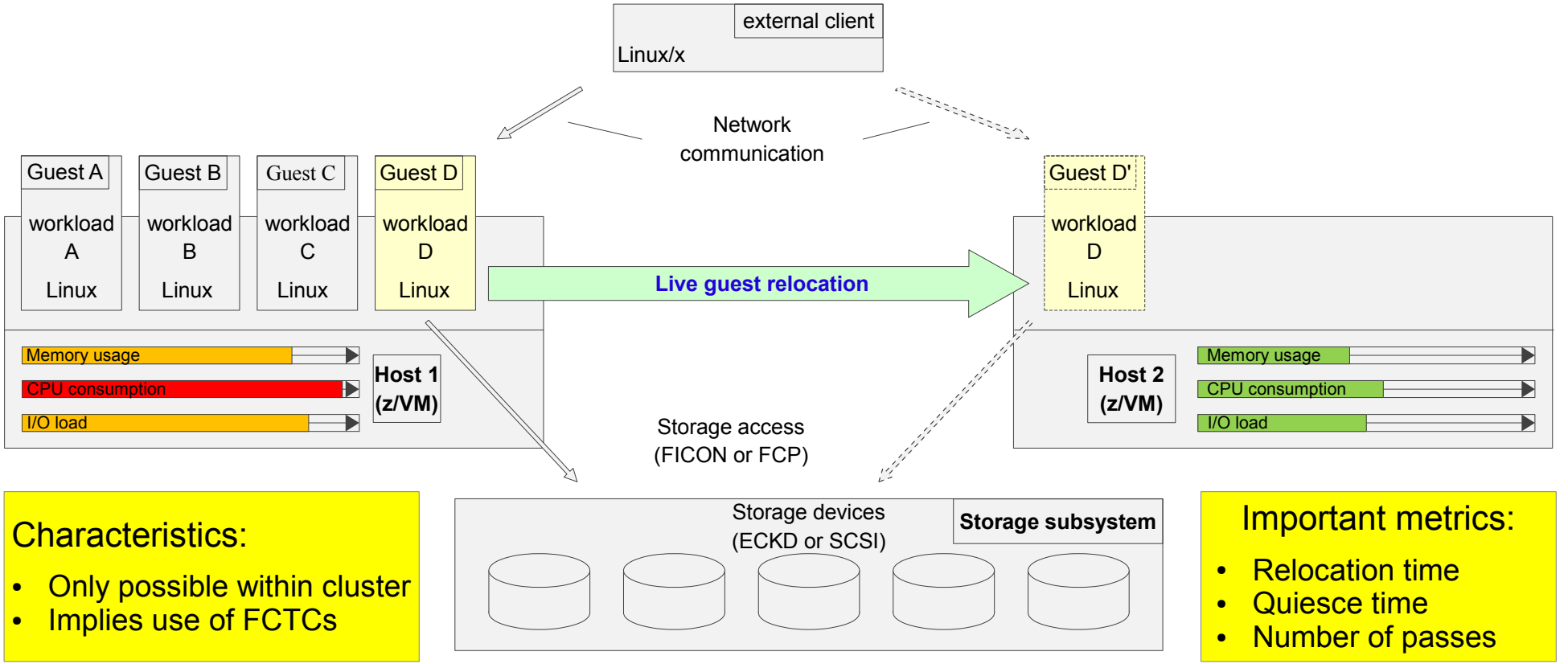
FICON
Fibre Connection

CTC
Channel-to-channel



- **Characteristics**
  - Two up to four SSI cluster members
  - One-to-one ISFC logical connections based on FICON CTCs between all members
  - Shared system configuration and spool – non-shared paging DASDs
  - Application DASDs either shared or non-shared
  - Access to common network
  - Access to common SAN

New function with z/VM SSI: z/VM Live guest relocation



Characteristics:

- Only possible within cluster
- Implies use of FCTCs

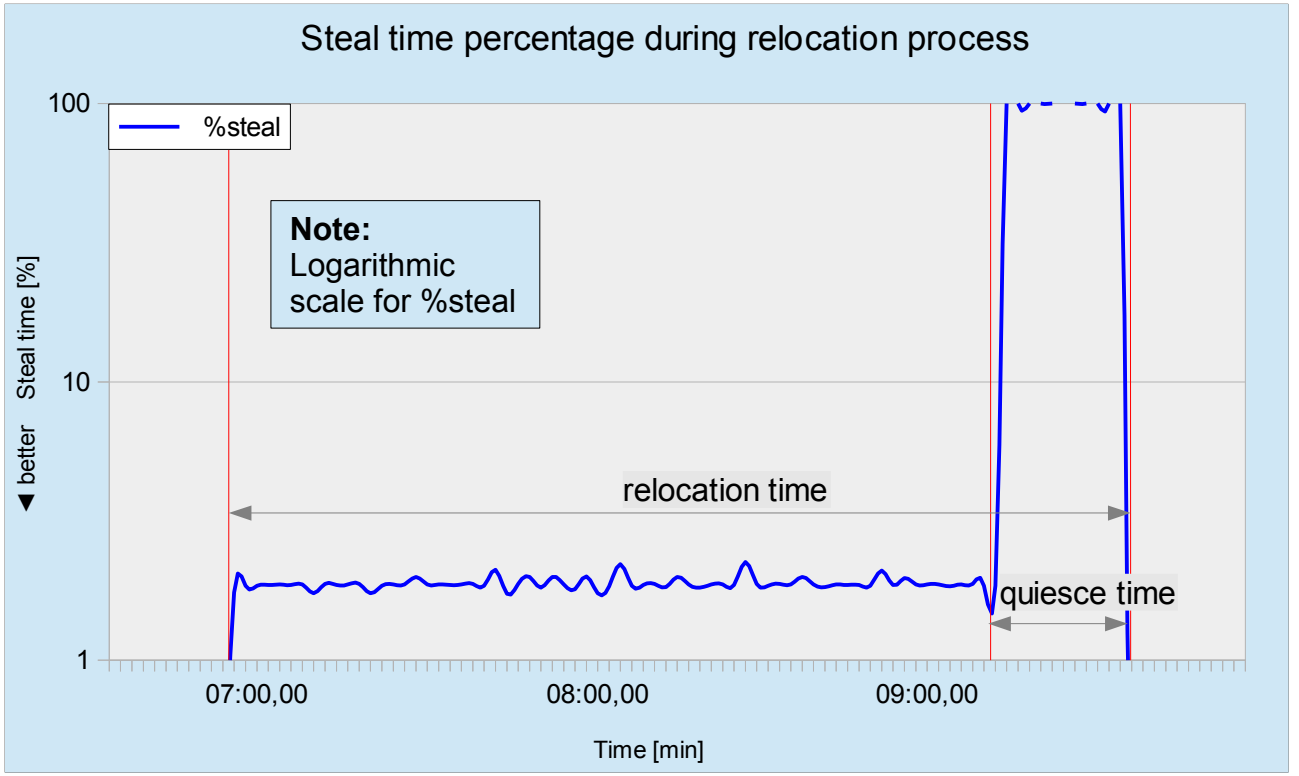
Important metrics:

- Relocation time
- Quiesce time
- Number of passes

- **Live guest relocation**
  - Provides for the relocation of a virtual system from one z/VM SSI cluster member to another
  - Guest continues operation most of the time while the transfer progresses
  - Relocation is performed by several scan passes over the guest's memory, transferring pages changed in the previous pass
  - Eventually, the guest is stopped ("quiesced"), and remaining changed pages at that time are transferred
  - Finally, the transferred guest resumes its work on the target z/VM system

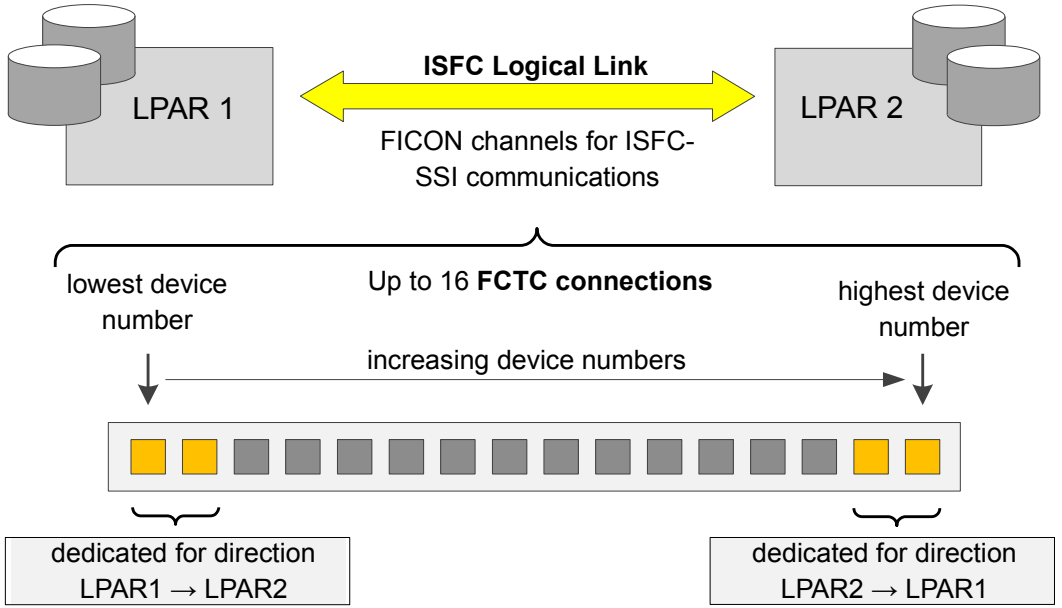
Relocation impacts observed by a Linux guest

Definition:
Steal time is the percentage of time a virtual CPU waits for a real CPU while the hypervisor is servicing another virtual processor.



- **Most significant Linux parameter is the steal time (I prefer "steal percentage")**
  - As the relocation starts, the slightly higher steal percentage reflects z/VM's use of resources for the relocation process
  - During the quiesce time, the Linux guest is not dispatched, resulting in 100 % steal percentage
  - Once relocated, the steal percentage falls back to the initial low value because on the new z/VM host also sufficient resources are available

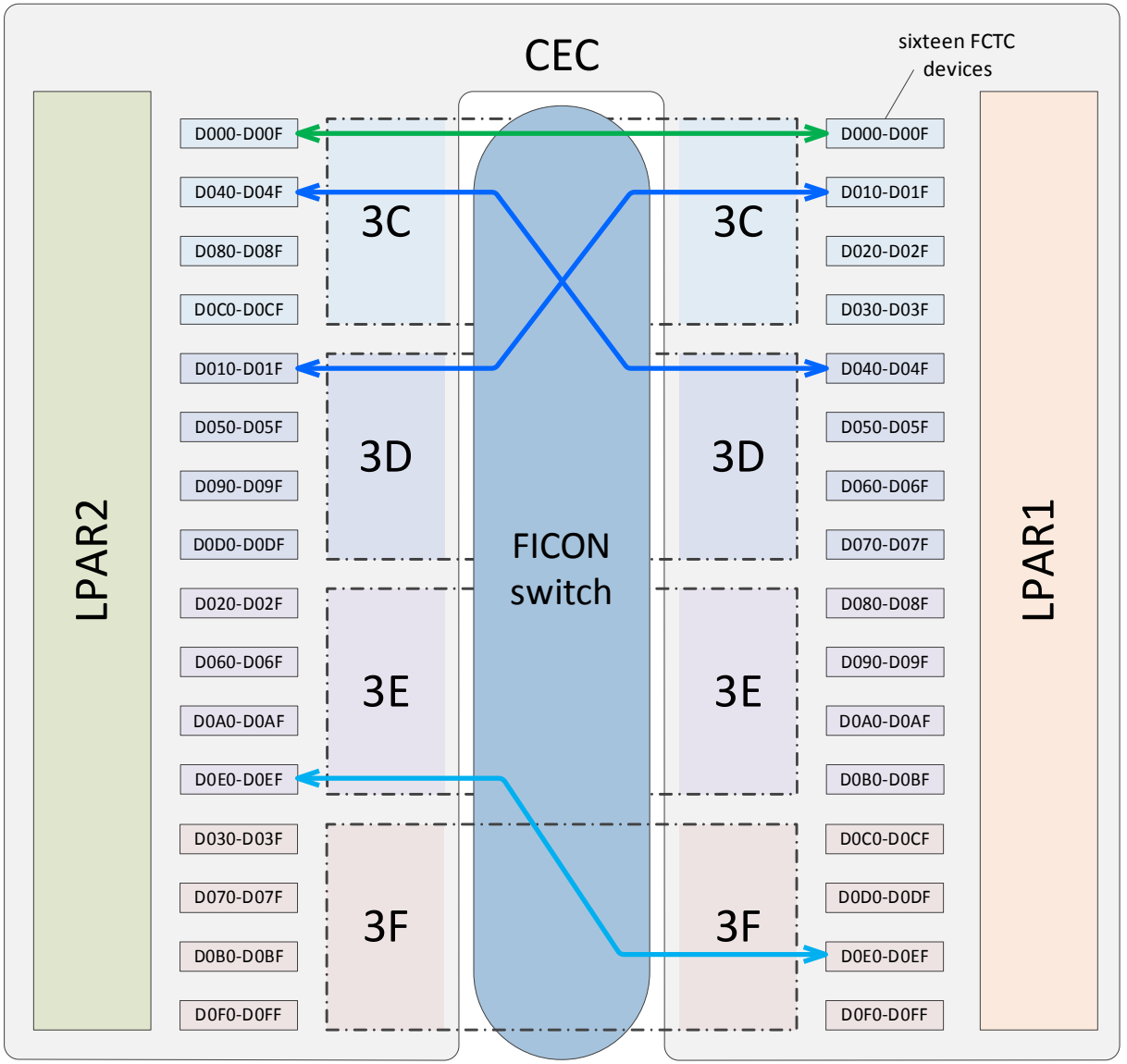
FCTC device configuration for an ISFC logical link



- In the IOCDS, it is practical to define more FCTC connections than actually needed, unless there are other conditions that rule against this (such as device number shortages)
- In z/VM, you can then optimize when defining the ISFC logical links by selecting from that set

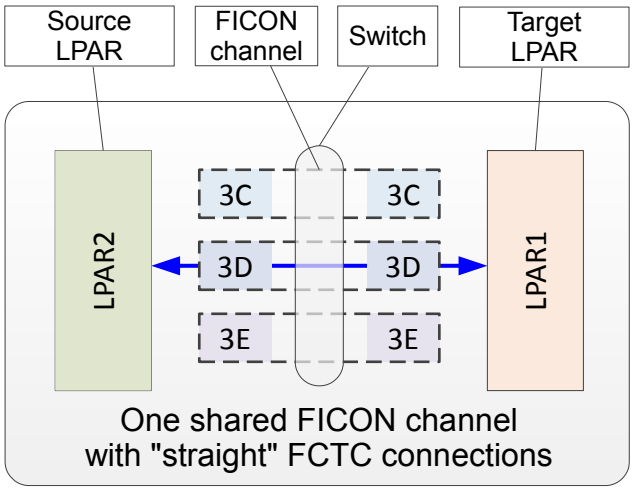
- **ISFC logical link**
  - An ISFC logical link connects two SSI cluster members (z/VM instances) point to point
  - Each ISFC logical link is composed of up to sixteen FCTC connections
  - Each FCTC connection consists of two FCTC devices
  - FCTC devices and their connections are defined in the IOCDS
    - FCTC devices are defined as usual, in scope of FCTC control units and FICON channels
  - z/VM dedicates certain FCTC connections for **unidirectional** use
    - For 2 – 8 FCTC connections: One FCTC connection is dedicated in each direction
    - For 9 – 16 FCTC connections: Two FCTC connections are dedicated in each direction
  - Dedication of uni-directional FCTC connections is based on z/VM host name and device number

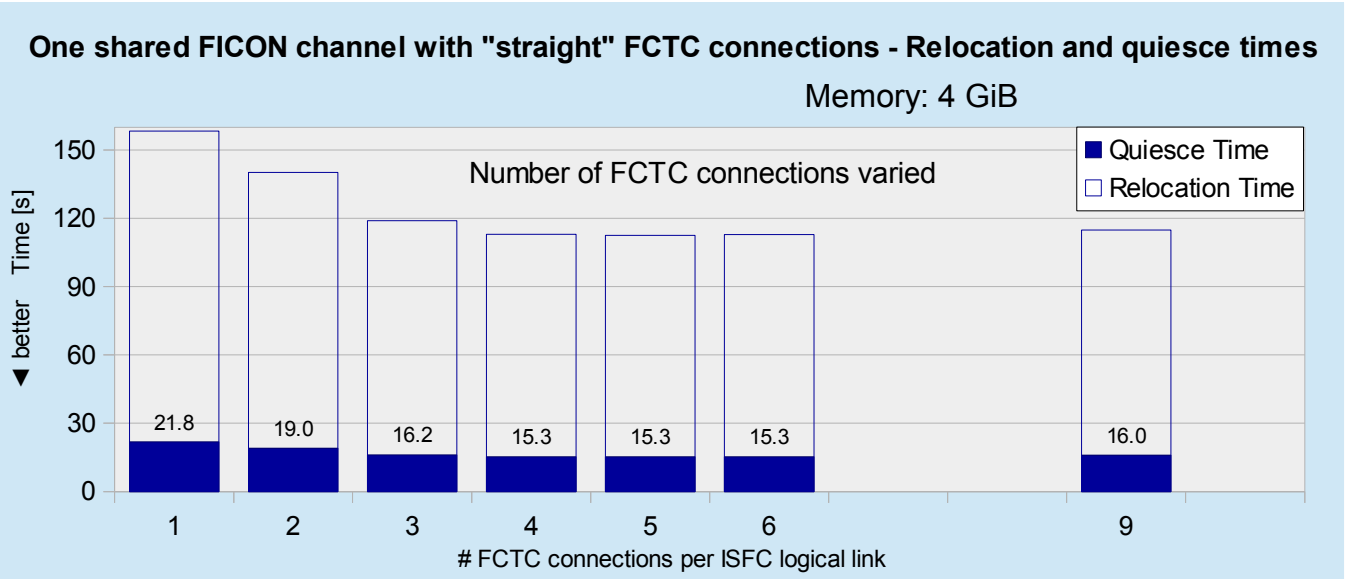
Example FCTC configuration with four FICON channels



- **Characteristics of this example:**
  - One CEC
  - Two LPARs (on that CEC)
  - Four FICON channels
  - All FICON channels connected to the same FICON switch
  - Each line reflects sixteen FCTC connections
- **Two kinds of FCTC connections:**
  - Straight: connecting FCTC devices on the same FICON channel
  - Crossed: connecting FCTC devices on different FICON channels
- **NOTE:**
  - In z/VM, the definition of an **ISFC logical link** can use up to **sixteen** FCTC connections (from those respectively defined in the IOCDS)
  - In our tests, different combinations of FCTC connections were used to define various ISFC logical link configurations

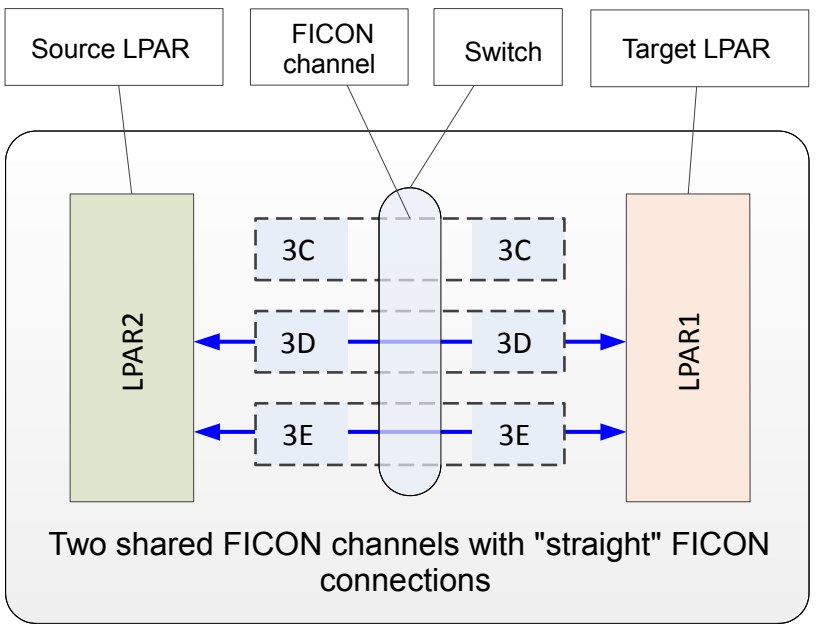
Relocation of a Java workload – one FICON channel

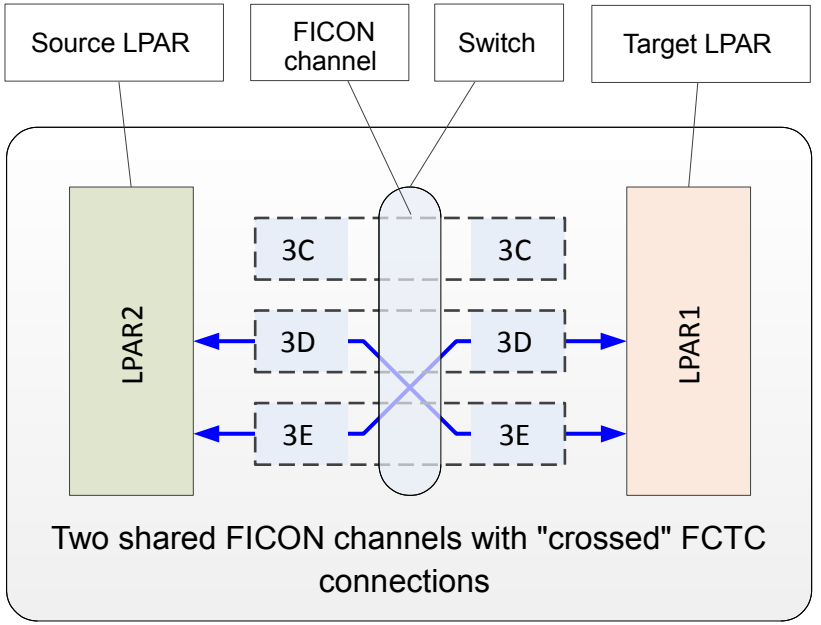




- **Scaling #FCTC devices on one FICON channel**
  - The relocation and quiesce times decrease as up to four FCTC connections are used
  - The relocation and quiesce times do not further improve as more than four FCTC connections are used
- **Conclusion**
  - Four FCTC connections seem sufficient to fully load a FICON channel
  - The improvement from using the fourth FCTC connection is small
  - z/VM recommendation: Normally, use four FCTC connections per FICON channel

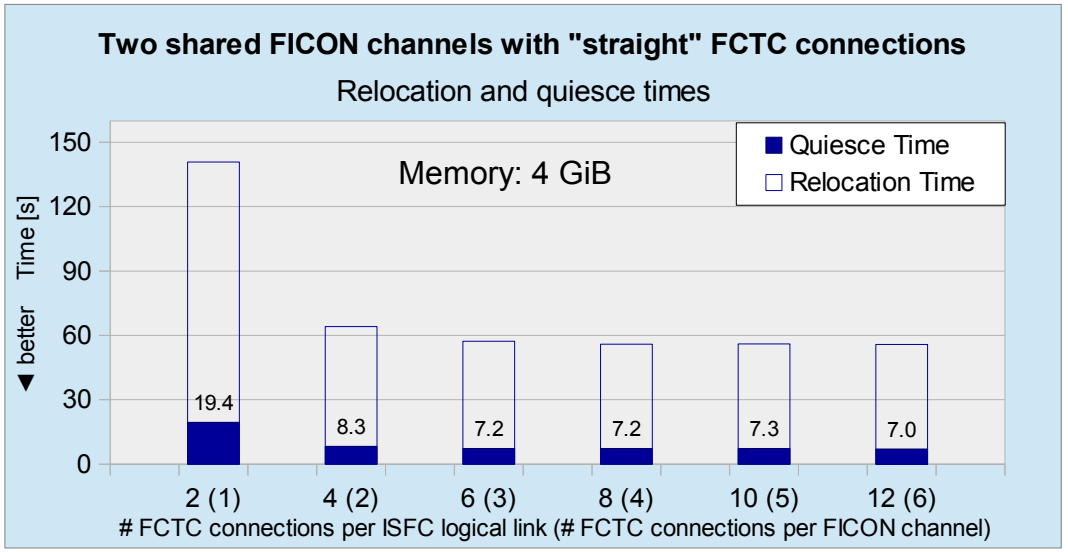
Configuration variants with two shared FICON channels

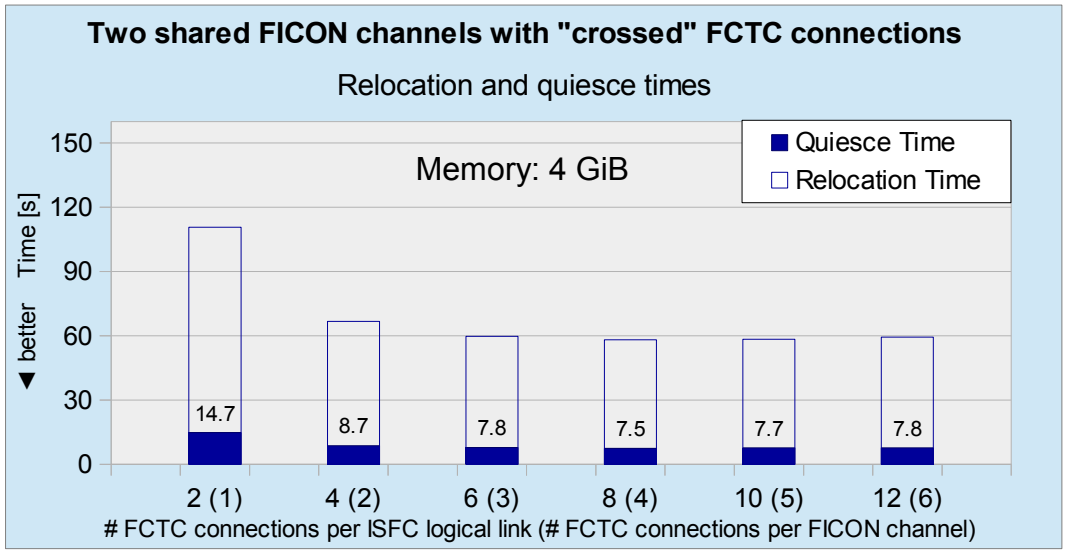




- **#FCTC devices on two FICON channels**
  - Variant I: "Straight" FCTC configuration
    - The FCTC devices on both ends are configured on the same FICON channel
  - Variant II: "Crossed" FCTC configuration
    - The FCTC devices on both ends are configured on different FICON channels
- **These variants apply only for FCTC connections within one CEC**
  - In both cases, the FICON channels are simultaneously used for sending and receiving data
  - The FICON channels are shared between the LPARs

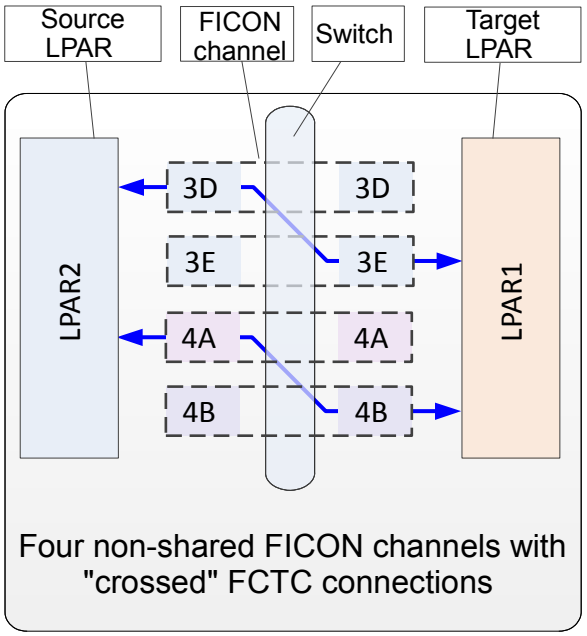
Relocation results: Scaling # FCTC devices on two shared FICON channels

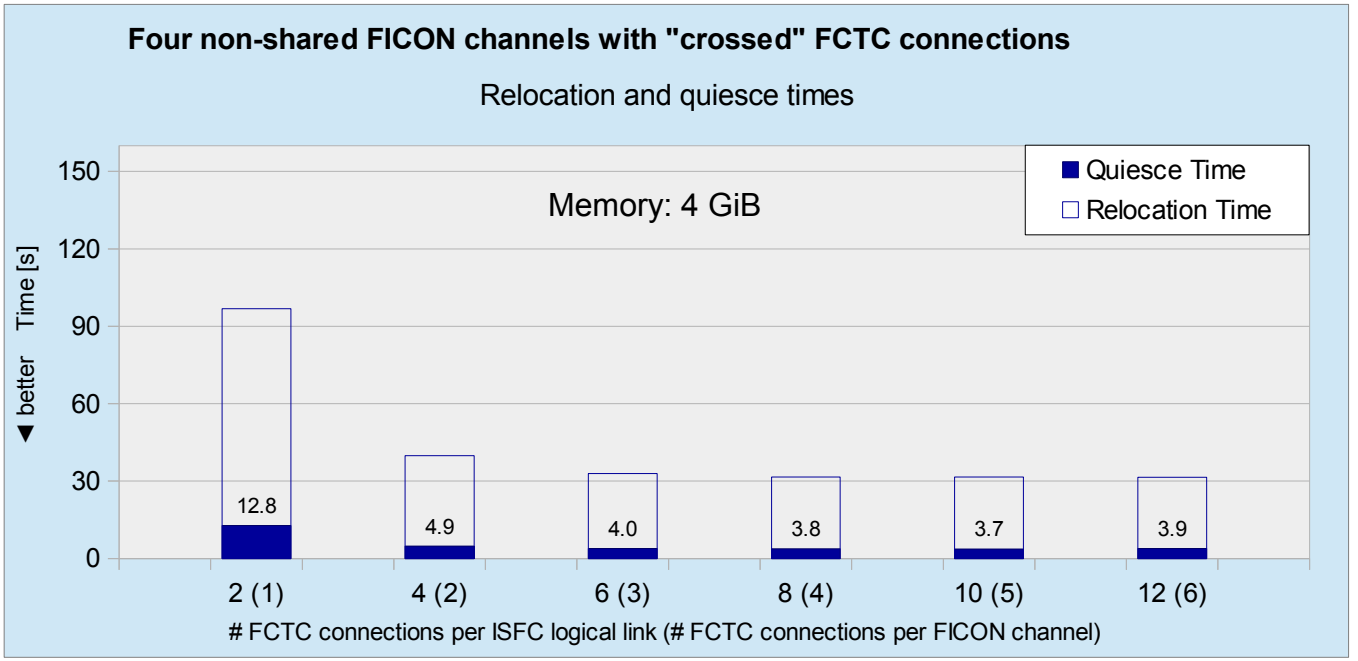




- **Observation**
  - Relocation and quiesce times decrease up to three FCTC connections / FICON channel
  - With two FCTC connections
    - one FCTC connection is reserved for unidirectional use
    - the relocation and quiesce times are lower for the "crossed" case
  - With four FCTC connections (and more)
    - the relocation and quiesce times become lower for the "straight" case
- **Conclusion**
  - Using three FCTC connections in this situation seems sufficient to load the FICON channels
  - With a crossed configuration, the FCTC connections used for data transfer send on one and receive on the other FICON channel

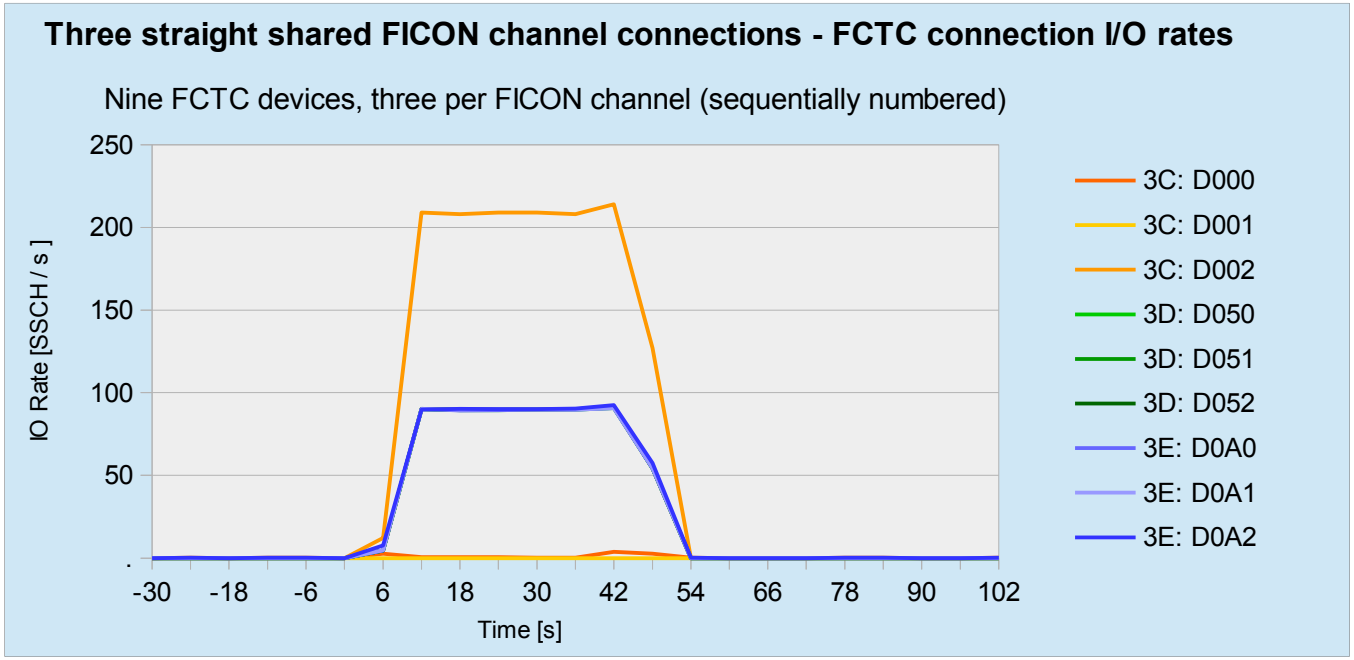
CTC Setup – non shared FICON channels





- **Observation**
  - The relocation and quiesce times decrease as one up to four FCTC connections are used
  - Each LPAR exclusively uses two FICON channels
- **Conclusion**
  - The relocation times are about half as long as in the "shared" case
  - In this "non shared" case, now also the fourth FCTC connections per FICON channel slightly improves the relocation behavior
- **NOTE**
  - This configuration closely resembles the case when the LPARs reside on two different CECs.
  - The data rates (not shown) on the FICON channels are about twice as high as in the "shared" case

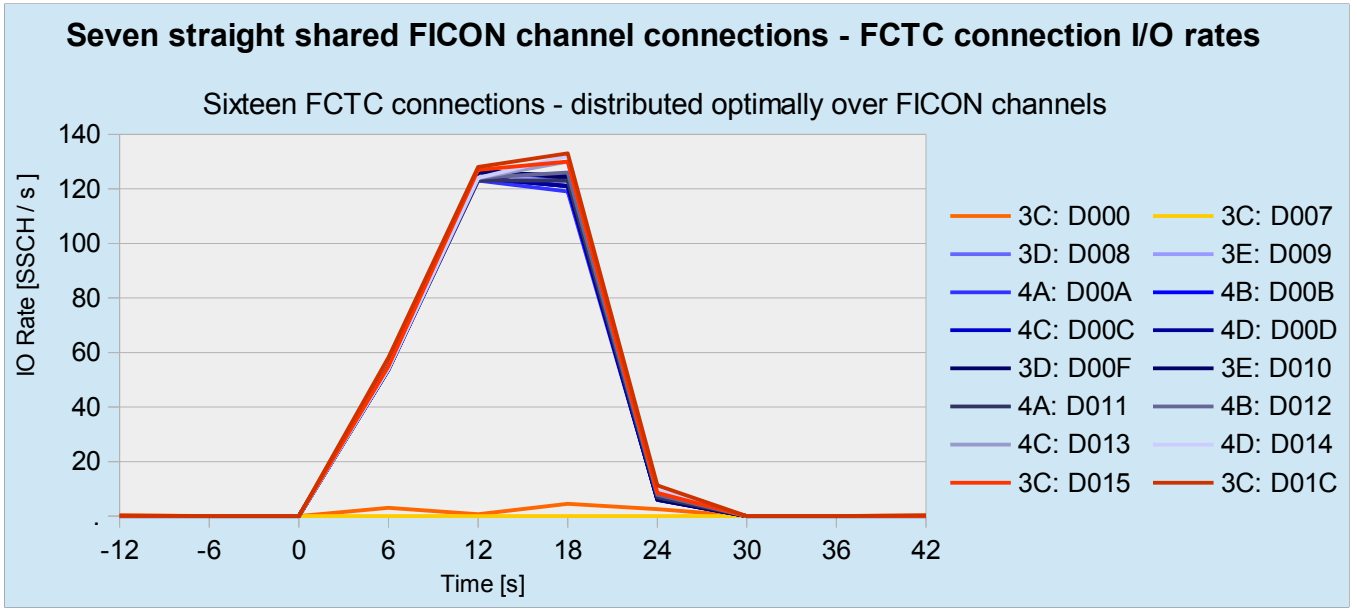
Impact of CTC connections dedicated for unidirectional use



- **IO rates on**
  - FCTC connections on 3D (green) and 3E (blue) process about 90 SSCH/s (270 SSCH/s per FICON channel)
  - FCTC connections on 3C differ (214 SSCH/s per FICON channel):
    - D000 (red) and D001 (yellow) exhibit almost zero IO rates
    - D002 (orange) exhibits 214 SSCH/s
- **Conclusion**
  - z/VM dedicates two FCTC connections for unidirectional use
    - D000 and D001 are dedicated for the direction target → source
    - unused for transferring data from source → target
  - Only one FCTC connection of FICON channel 3C is used → operates significantly below its capacity

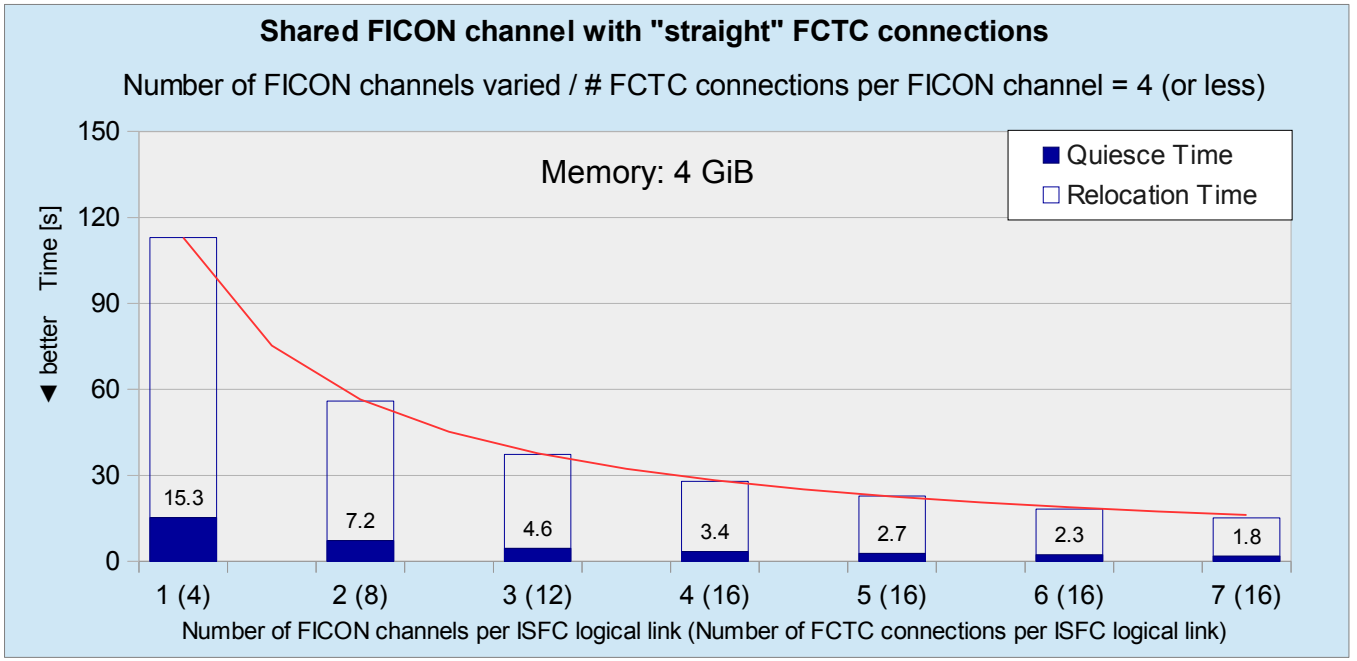
Avoid configurations where on one FICON channel less than three FCTC devices are used for data transfer in either direction

Optimized FCTC Setup



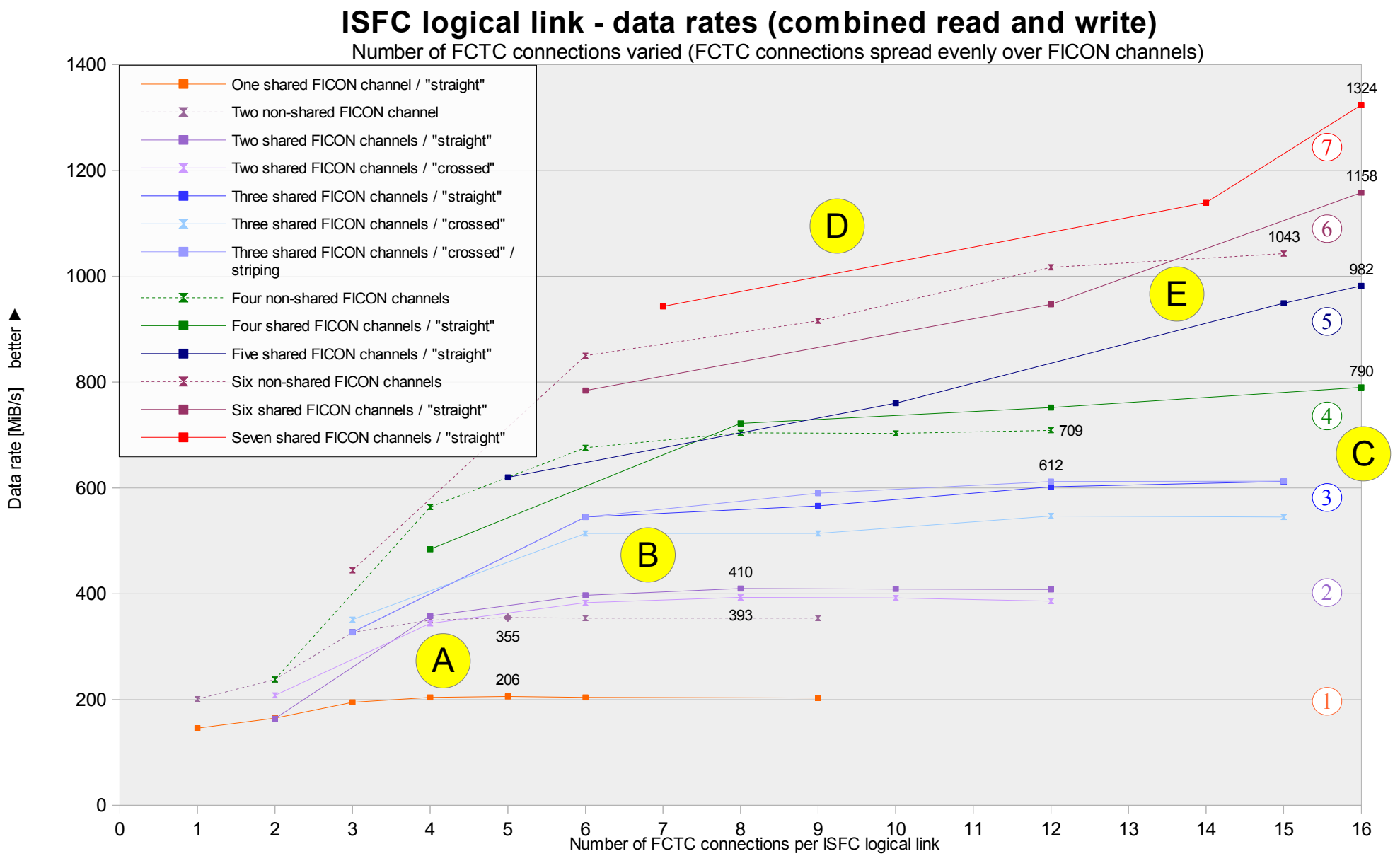
- **ISFC logical link configuration (16 FCTC connections overall, 7 FICON channels)**
  - Two FCTC connections per FICON channel, except FICON channel 3C with four FCTC devices
  - In each direction, z/VM dedicates two FCTC connections for uni-directional use
- **Aim for each FICON channel having the same amount of <u>active</u> devices**
  - Configure the first two and the last two FCTC devices on one FICON channel (here 3C)
  - So on that FICON channel
    - two FCTC devices are uni-directional in one direction
    - the other two FCTC devices are uni-directional in the other direction
    - so out of the four configured FCTC devices, two are used for data transfer in either direction

Scaling FICON channels in a shared setup



- **Scaling FICON channels**
  - The relocation and quiesce times scale about inversely linear with the number of FICON channels
- **Conclusion**
  - Adding FICON channels is a main parameter for improving the relocation and quiesce times
  - The number of FCTC connections for an ISFC logical link is limited to a maximum of sixteen
    Then two FCTC connections at each side are used in one direction only
    - Thus, the use of more than seven FICON channels for an ISFC logical link is uneconomic except maybe if the FICON channel(s) are used for other purposes as well

ISFC logical link data rates



ISFC logical link data rates

- **What can be learned by observing the data rates ?**

A – A FICON channel is kind of "saturated" with four or more FCTC connections
  - Conclusion: Use four FCTC connections per FICON channel

B – Typically there is no significant difference between straight and crossed configurations
  - Conclusion: Use whatever is easier to configure
  - However: Ensure that uni-directional devices are not all on one FICON channel

C – Data rates roughly scale with the number of FICON channels
  - Conclusion: Use as many FICON channels as you can afford
  - **BUT:**

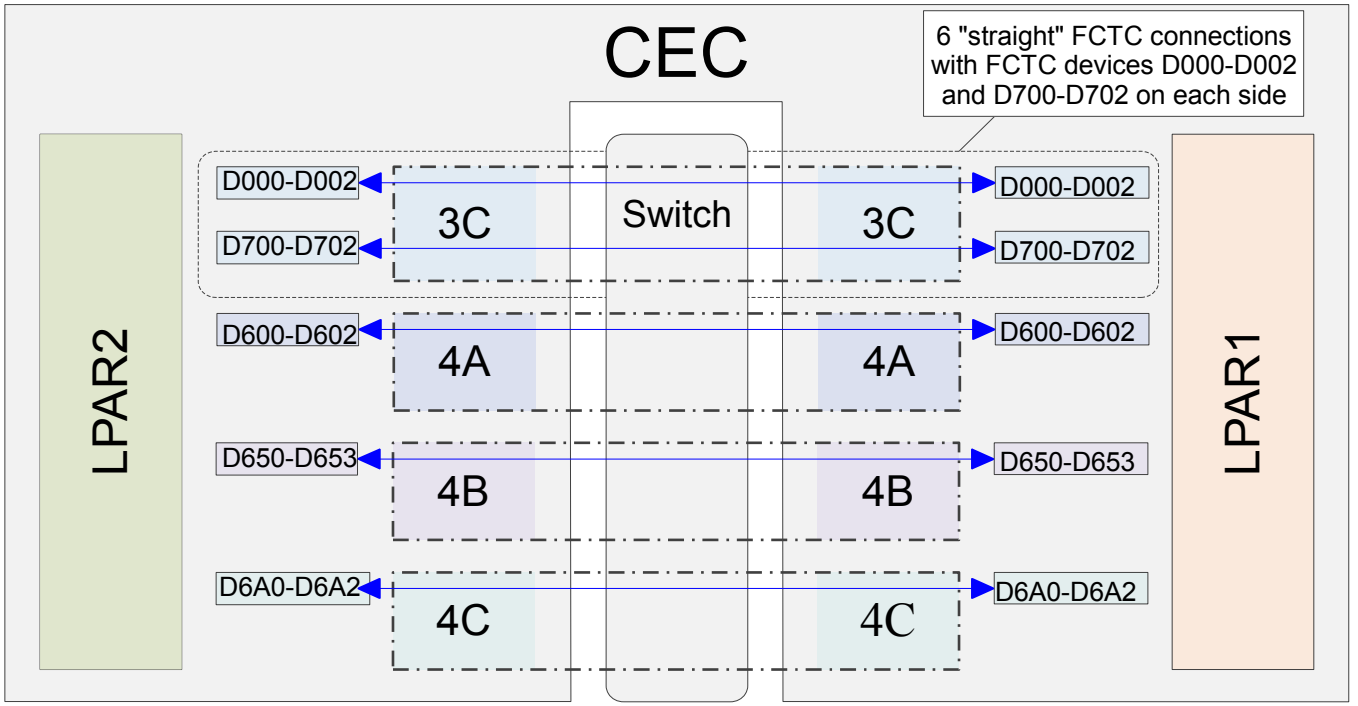
D – As more than four FICON channels are used, less than four FCTC connections per FICON channel result because an ISFC logical link is limited to 16 FCTC connections
  - Conclusion: When more than four FICON channels are used, it is not possible to "saturate" all FICON channels
  - Possible approach: Consider using the FICON channels for other purposes as well (however, this is not encouraged in z/VM documentation)

E – The "non-shared" configurations produce slightly higher data-rates when less than four FCTC connections per FICON channel are used, and slightly lower data-rates when four or more FCTC connections per FICON channel are used
  - Possible Explanation: The "shared" configurations use the FICON channels in both directions and thus afford twice the number of FCTC devices per FICON channel as opposed to the non-shared case – so in the non-shared case, when less than four FCTC devices are used, a channel is not yet "saturated"

PART II

Linux guest and application considerations

Test Configuration: Optimized FCTC setup using four FICON channels

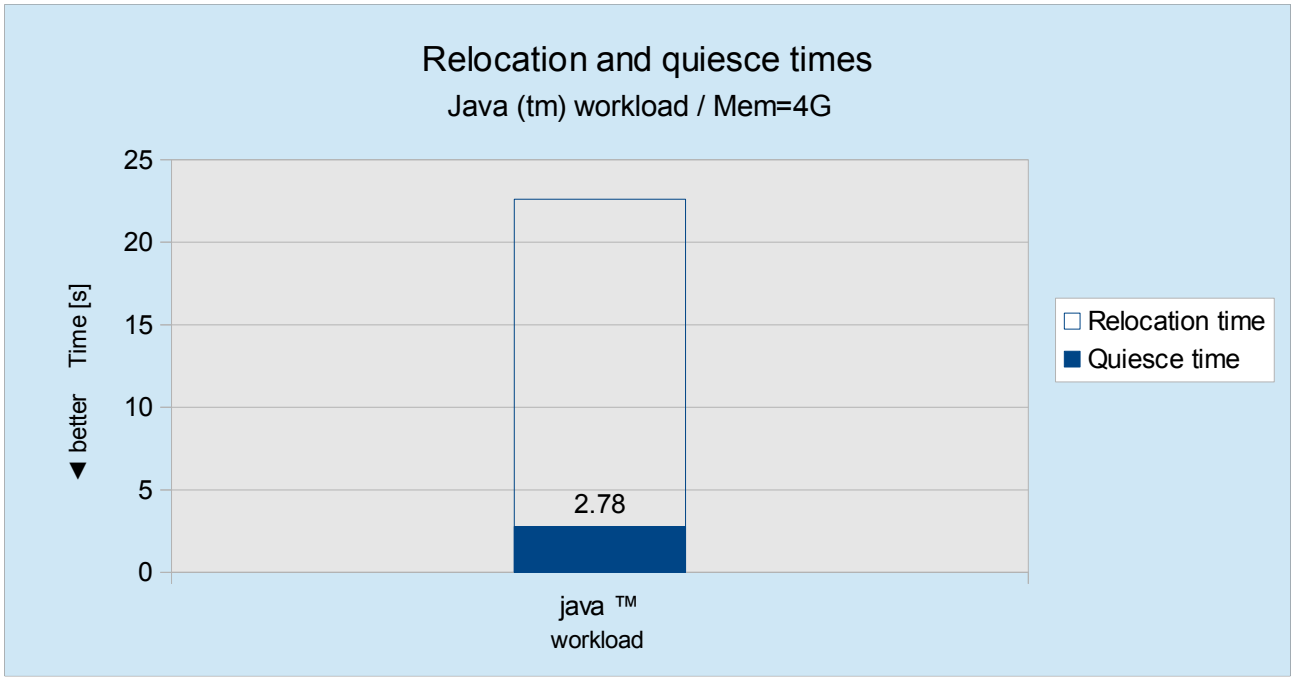


- **Optimized ISFC logical link configuration using four FICON channels**
  - Four shared FICON channels, using the "straight" setup for FCTC connections
  - FICON channel 3C: 6 FCTC connections
  - including the two lowest and the two highest devices (→ 4 "active" FCTC connections in either dir.)
  - FICON channel 4A and 4C: 6 FCTC connections (3 FCTC connections each)
  - FICON channel 4B: 4 FCTC connections
  - As a result, for data transfers in either direction, always fourteen out of sixteen FCTC connections used for data transfer, with at least three active FCTC devices per FICON channel

Test scenarios

- **Three different kinds of workload**
  - Java™ workload
    - Reference workload from part I
  - Filesystem workload
    - Variation: impact of the use of the page cache
  - Transactional database workload
    - Variation: scaling virtual guest sizes
- **Two test scenarios**
  - The relocation behavior of each individual workload is analyzed separately
  - All three workloads are started on the source system
    - The transactional database workload is relocated and its relocation behavior is analyzed
- **Compare relocation metrics**
  - Relocation time
  - Quiesce time
  - Number of memory transfer passes

Reference workload: The Java™ workload used in Part I

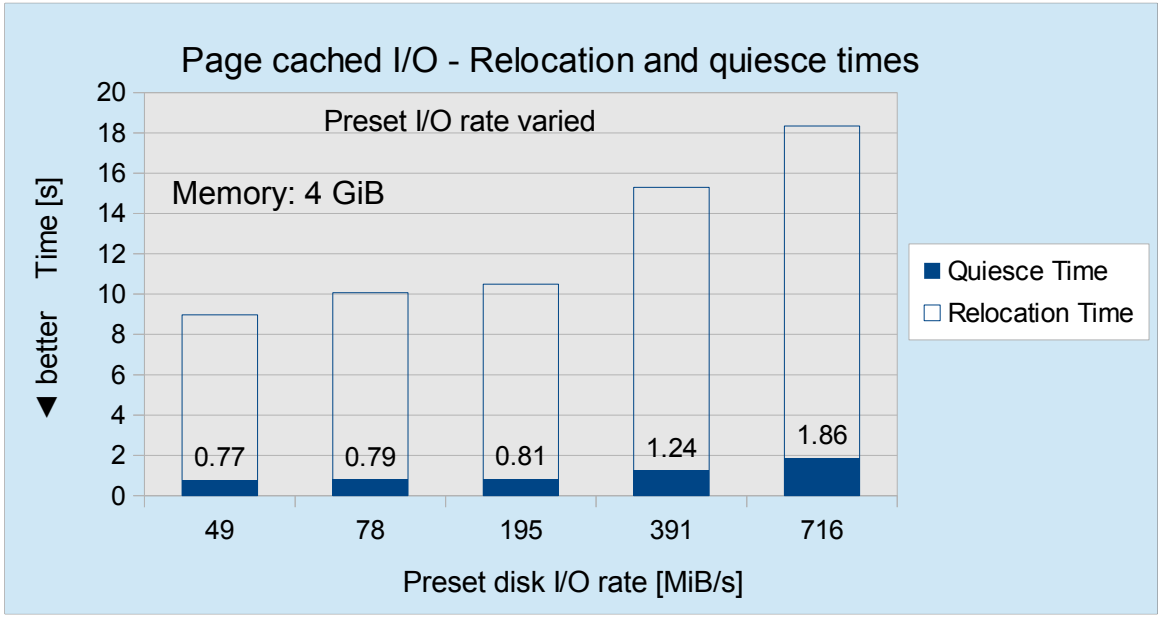


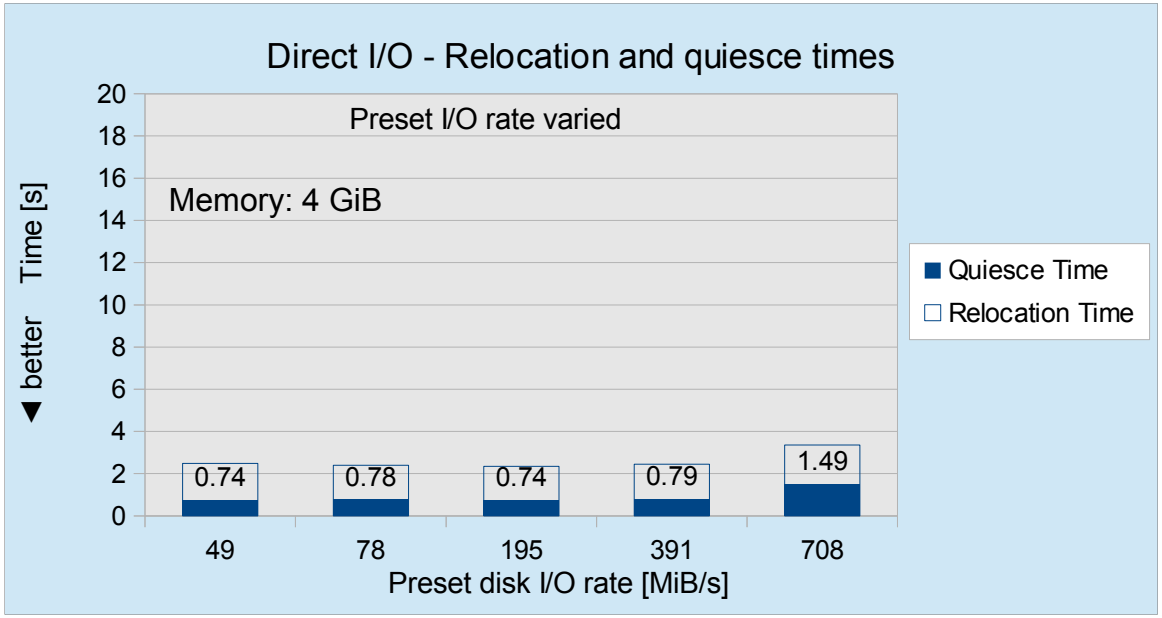
- **The Java™ workload analyzed in Part I is used as reference workload**
  - The virtual system size is 4GiB
  - One virtual processor
- **Factors influencing the relocation process**
  - Java heap processing
  - Amount of memory pages modified per second ("page dirty rate")
- **It is expected that other Java™ workloads behave similar**
  - For example, WebSphere Application Server

Filesystem workload (disk I/O)

- **Pure random write workload**
- **Factors influencing the relocation process**
  - Type of I/O: page-cached or direct
  - I/O configuration: external disks, HyperPAV devices, FICON channel
  - Amount of memory pages modified per second, influenced by
    - Configured preset I/O rate
    - Configured application buffers (size fixed)
    - Configured I/O subsystem buffers
    - Configured page cache usage
      - Application buffers are also written to the page cache, controlled by the preset I/O rate
      - Additionally, free memory page frames (not in use otherwise) are used as page cache

Filesystem workload (disk I/O) - Results



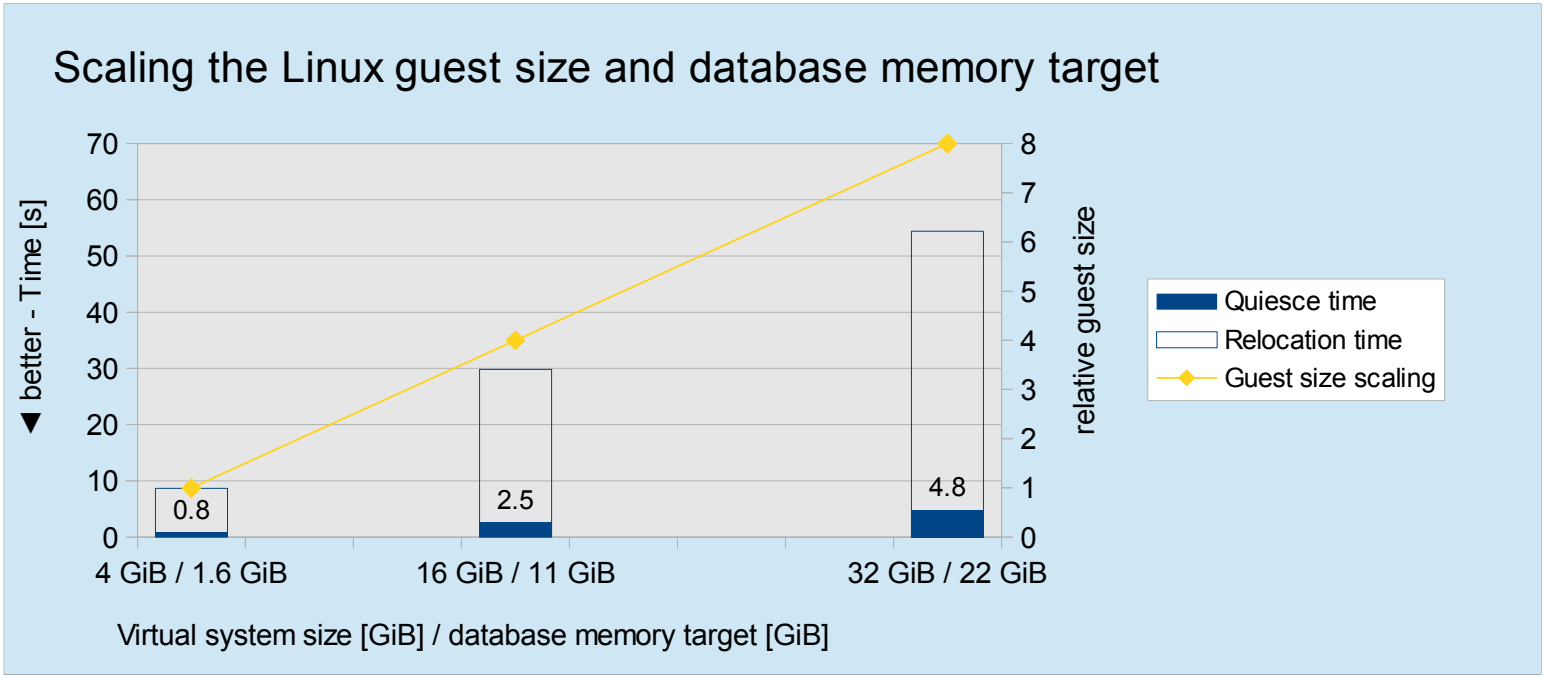


- **Page cached I/O**
  - Relocation and quiesce times rise as the preset I/O throughput is increased
- **Direct I/O**
  - Relocation times are much lower than for the page cached I/O case
  - Quiesce times are a little bit lower
  - Workload throughput does not significantly affect both values, except at the highest (unconstrained) level
- **Conclusion**
  - Workloads doing direct I/O require less time to relocate than those with page cache I/O
  - During the quiesce time, there seems to be a constant base effort for z/VM that does not depend on the workload's memory modification rate

Transactional database workload (OLTP)

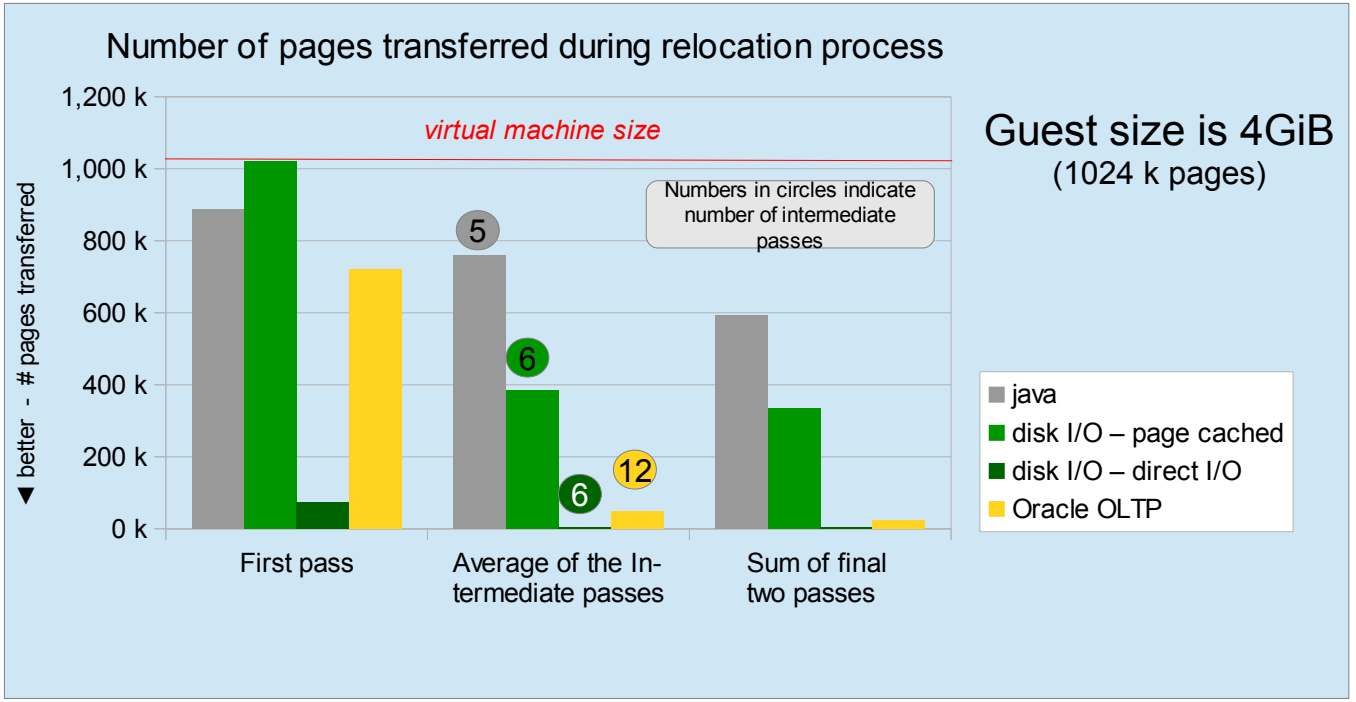
- **CPU intensive (4 CPUs), random read and write disk workload**
  - Scaling system size and database buffer sizes
  - Using direct I/O and asynchronous I/O
  - External clients acting as work load generators
- **Factors influencing the relocation process**
  - Number of I/O devices used: 4 FCP adapters
  - Database buffer pools: Assigned around 70% of memory available memory
  - Amount of memory pages modified per second ("page dirty rate")
- **Memory modification rate**
  - Database buffers (controlled by database software, usually fixed size)
  - Direct I/O - no page cache → reduced use of memory
  - Databases are very effective caching systems → efficient use of memory

Transactional database workload - Results



- **Relocating a transactional database under an transaction workload (CPU intensive, ~ 100 MB/sec read + write)**
- **Scaling Linux guest size and database memory target (4 CPUs)**
  - The quiesce times scale lower than the memory size
  - The relocation times scale higher, but still less than the memory size

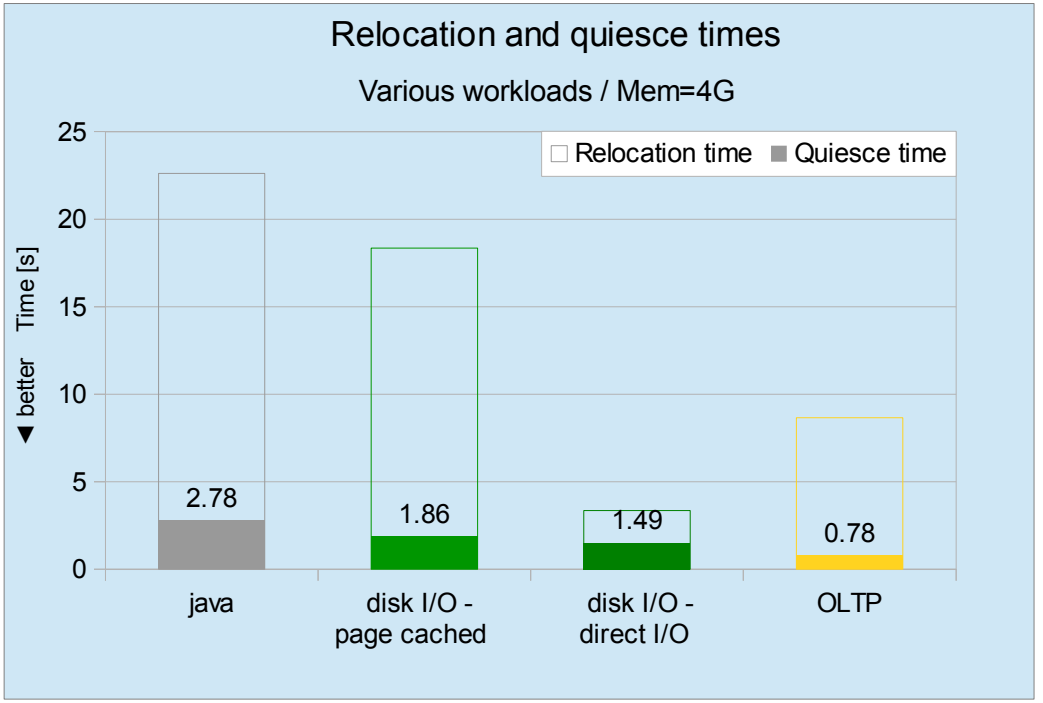
Workload comparison I



- The amount of memory transferred during the first pass is determined by the guest's "past"
- The amount of memory transferred during subsequent passes is determined by the guest's memory change activity during the relocation process
- The number of passes depends on how well the workload's memory change activity decreases as the relocation progresses

- **Amount of passes and pages transferred varies with the kind of workload**
  - The Java™ workload exhibits the lowest amount of passes, the highest amount of transferred pages, and the decrease during subsequent passes is low
  - The transactional database workload exhibits the highest amount of passes, the decrease of pages transferred per pass is also the highest (the center columns only reflect the average ...)
  - The filesystem workload with page cache uses the whole guest size, but the amount of pages transferred during subsequent passes is fairly reduced, but still a signification amount must be transferred during the final passes
  - The filesystem workload with direct I/O has the lowest memory requirements

Workload comparison II

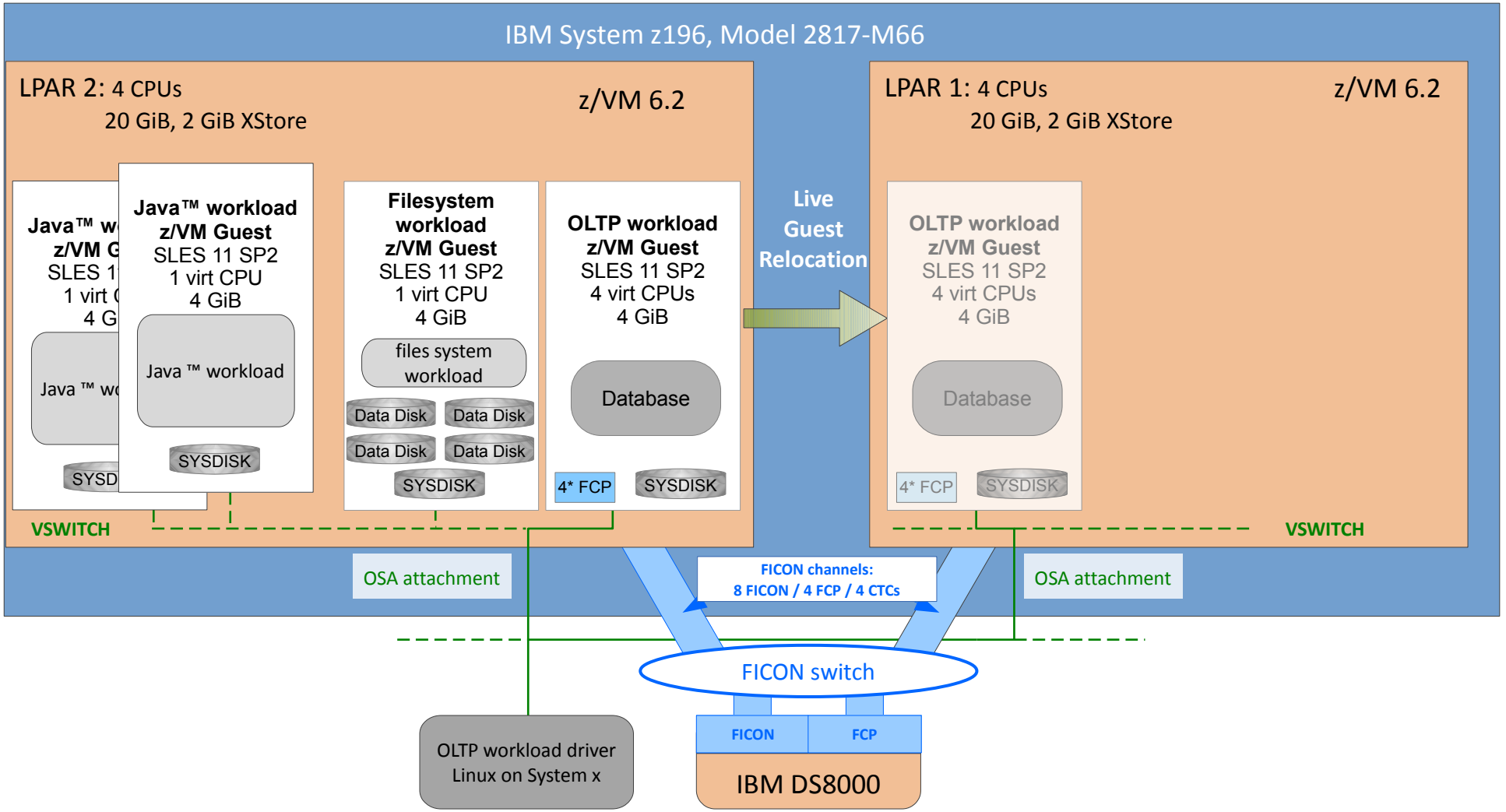


- **Java has the highest time values**
  - A constantly high amount of pages written in memory transfer passes
  - Possible explanation: Java memory handling and garbage collection
- **File system workload with page cache**
  - High total relocation times
  - Quiesce time is close to the direct I/O case
- **File system workload with direct I/O**
  - Causing the lowest relocation effort
  - Quiesce time is relatively high in relation to the amount of memory pages transferred
    - There seems to be a certain base amount in the quiesce time required for other activities than transferring memory
- **The effort for the database workload (using direct I/O) is less than expected**
  - Moderate relocation time, but extremely short quiesce time
  - Memory changes between the various passes seem to be relatively local (i.e., on the same set of memory pages)

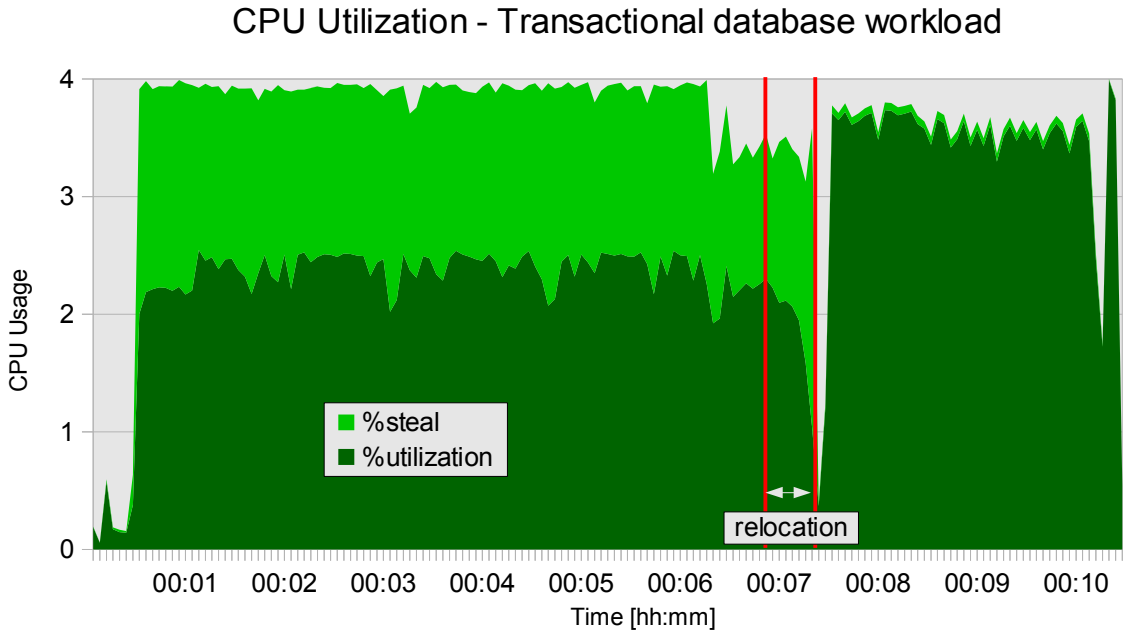
PART III

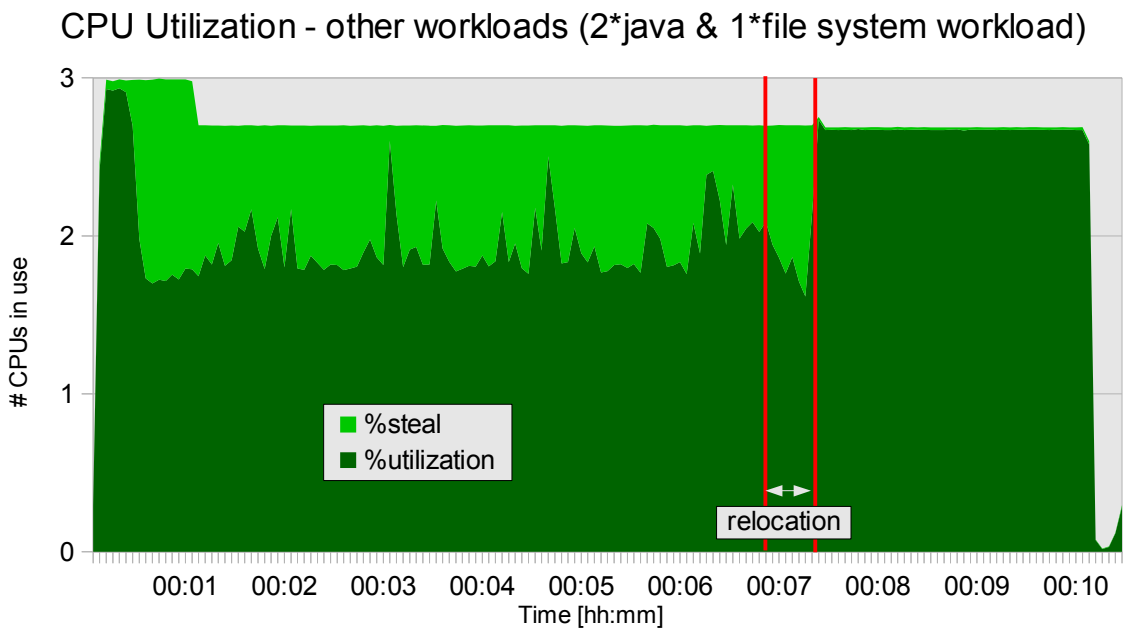
Resolving a resource constrained scenario with LGR

LGR - Resolving a resource constrained scenario I



LGR - Resolving a resource constrained scenario II





- **Transactional database workload**
  - 4 virtual processors configured
  - constrained before relocation
  - unconstrained after relocation
- **Other workloads (not relocated)**
  - Three workload instances
    - 2 * Java™
    - 1 * Filesystem workload
  - One virtual processor configured for each instance
  - Combined workloads utilize a little less then 3 processors
    - Idle / wait times result from filesystem workload
  - Constrained before DB workload relocation
  - Unconstrained after relocation of DB workload

Summary

- **The configuration of the ISFC logical link significantly impacts the relocation times**
  - Relocation times scale inversely linear with the amount of FICON channels
  - Configure four 4 FCTC devices per FICON channel (in some cases even 3 FCTC devices suffice)
  - The FCTC connections reserved for unidirectional use can influence the performance if not planned carefully
  - Relocation and quiesce times scale with the size of the used guest memory (but at a significantly lower rate)
- **Workload comparison**
  - Java workloads have relatively high relocation and quiesce times
  - The transactional database workload had the shortest quiesce times
  - Workloads with direct I/O have shorter relocation times than when the page cache is used
- **Moving workloads to resolve a resource issue**
  - The smaller the used memory of a guests the shorter the relocation and quiesce times
  - The database workload was a good first candidate (i.e., high workload, but low relocation times)
  - The Java workloads imposed relatively high effort for the relocation process
  - The impact observable by workload end-users is directly related to quiesce time
  - The important parameter for the administrator is the relocation time
    - observing the recommendation to relocate only one guest at a time

Questions ?

- **Further information is located at**
  - z/VM 6.2 Live Guest Relocation with Linux Middleware
    http://www.ibm.com/developerworks/linux/linux390/perf/tuning_vm.html#grel
  - z/VM 6.2 Live Guest Relocation with Linux Middleware - Various Loads
    http://www.ibm.com/developerworks/linux/linux390/perf/tuning_vm.html#grel2
  - Linux on System z – Tuning hints and tips
    http://www.ibm.com/developerworks/linux/linux390/perf/index.html
  - Live Virtual Classes for z/VM and Linux
    http://www.vm.ibm.com/education/lvc/

IBM

Dr. Juergen Doelle

Linux on System z
System Software
Performance Analyst

IBM Deutschland Research & Development
Schoenaicher Strasse 220
71032 Boeblingen, Germany

IBM

Michael Johanssen

Linux on System z
System Software
Performance Analyst

IBM Deutschland Research & Development
Schoenaicher Strasse 220
71032 Boeblingen, Germany