

May 2009



## **Performance of an Oracle 10g R2 Database Import Environment**

***Table of Contents***

Objectives ..... 2  
Environment setup ..... 3  
    Storage server ..... 3  
    Host ..... 3  
    Software..... 4  
    z/VM..... 4  
    Linux ..... 4  
    Oracle ..... 5  
The performance view of the import process ..... 6  
Architecture comparison..... 8  
Summary ..... 10

## Objectives

The Oracle's import and export utilities transfer data and structures stored in an Oracle database between multiple systems - independent of platform or product versions.

This paper<sup>1</sup>

- Describes the setup of an environment using the Oracle 10g R2 database on Linux<sup>®</sup> on IBM System z<sup>®</sup>
- Compares the performance of importing data into the database on the new IBM System z10<sup>™</sup> and on IBM System z9<sup>®</sup>.

This project emerged from a customer considering to migrate his core workload on a System z9 or a System z10.

- Currently, he is running some banking application workload on several distributed servers. He plans to migrate the core workload to Linux (Red Hat) on System z. The application performance had to be measured between System z9 versus System z10. The scenario is a data import from a DS8000<sup>™</sup> storage server into an Oracle 10g R2 database running Red Hat Linux under z/VM<sup>®</sup> 5.3 on a System z.
- The size of the environment in terms of CPUs, memory and disks, was defined according to customer needs.

<sup>1</sup> This paper is intended to provide information regarding performance of environments using the Oracle import utility and the Oracle database. It discusses findings based on configurations that were created and tested under laboratory conditions. These findings may not be realized in all customer environments, and implementation in such environments may require additional steps, configurations, and performance analysis. The information herein is provided "AS IS" with no warranties, express or implied. This information does not constitute a specification or form part of the warranty for any IBM products.

## Environment setup

This topic describes how we set up the environment using the Oracle 10g R2 database on Linux on IBM System z.

### *Storage server*

We configured a DS8000 2107-932 storage server with

- 16 FCP ranks and 256GB cache
- physical DDMs: 300 GB, 15k RPMS
- firmware level 63.0.106.0
- 4 GBits/sec FCP channel connectivity (2 channels were used)

### **Using striped FCP disks**

We created 3 SCSI volumes. For easy handling we configured them to a size of 300 GB. In the standard setup each of these disks would have been from a single rank, which would have been a significant performance bottleneck. The DS8000 firmware level allowed us to use a new feature: the *storage pool striping* function. To ensure maximum I/O bandwidth, we created one extent pool over the eight ranks on each of the two internal servers of the DS8000, and striped the volumes across all ranks of this extent pool. The storage server was connected to the System z via two 4 Gbit FCP channels.

### *Host*

For the tests we used one logical partition (LPAR) from each of these two systems:

1. IBM System z9 Enterprise Class (z9<sup>®</sup> EC), model 2094-S18 (1.7GHz),
2. IBM System z10 Enterprise Class (z10 EC<sup>™</sup>), model 2097-E26 (4.4GHz)

The LPAR used was equipped with 5 CPUs and 20 GiB central storage (see Table 1). The remaining hardware was not used for this test. z/VM was set up in this LPAR providing a virtual environment (guest) to run the database server as a Linux guest. The Linux guest itself uses a subset of the resources available to z/VM (see Table 1) to ensure that the virtualization causes no contention.

Table 1. System z LPAR and guest definitions

<b>System</b>	<b>Type</b>	<b>CPU</b>	<b>Memory size [GiB]</b>
z/VM	System z LPAR	5	20 GiB central storage + 2 GiB expanded storage
Database server	z/VM guest	2 (virtual)	16 GiB (virtual memory)

## *Software*

Table 2. Software levels

<b>Product</b>	<b>Version/Release</b>
z/VM	5.3
Red Hat Enterprise Linux	4.5
Oracle database server	10.2.0.2

## *z/VM*

For the z/VM setup, we disabled the queue-I/O Assist for the FCP adapters in the user's directory entry with

```
DEDICATE xxx yyyy NOQIOASSIST
```

## *Linux*

We increased the values for shared memory setup via `sysctl.conf` :

```
kernel.sem=250 32000 100 128
net.ipv4.ip_local_port_range = 1024 65000
fs.file-max=65536
```

Increasing the amount of TCP/IP ports and the number of file handles is a standard recommendation. The shared memory limits were set high enough to ensure they did not restrict the size of the Oracle buffer pools.

The file systems were set up with separate disks for either database data files, database log files, and the import file. This setup preserved the sequential character and direction of the I/O streams and kept them away from the bidirectional randomized disk I/O pattern of the disk storing the database data files (see Figure 1).

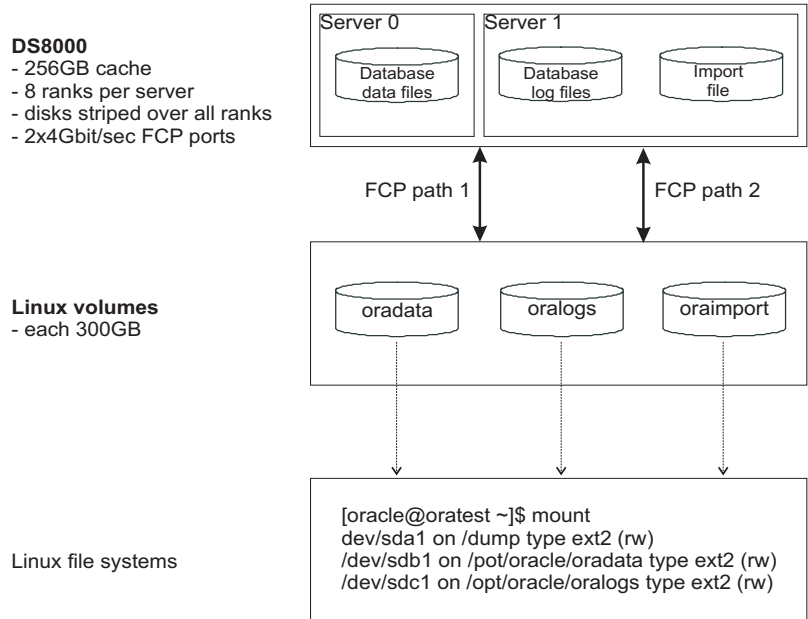


Figure 1. Disk organization of the import environment

*Oracle*

We configured the Oracle database with the following profile:

```
#####
#####
# initloadtest.ora
#####
#####

db_name=loadtest
db_files = 500
db_file_multiblock_read_count = 8
log_checkpoint_interval = 10000
processes = 512
parallel_max_servers = 32
log_buffer = 32768
max_dump_file_size = 10240
undo_management = auto
global_names = TRUE
filesystemio_options = setall
sga_target = 12582900012
undo_retention = 0
```

We used the asynchronous and direct disk I/O, and specified an SGA target to let the database automatically tune the proper buffer sizes.

**The performance view of the import process**

From the performance perspective, the import process is a mechanism with a complex structure, as depicted in Figure 2.

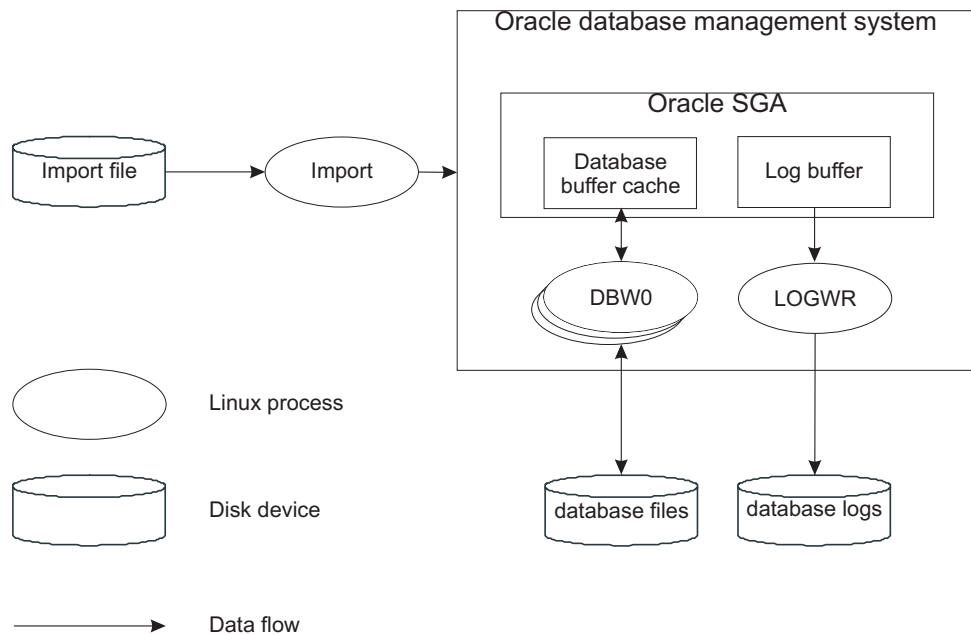


Figure 2. Structure of the import process

1. The procedure starts with the Oracle Import process. This import process is a single threaded application, so it does not benefit from multiple processors. This process reads a certain chunk of data from a file, prepares the data, and transfers them to the database.

This pure sequential read disk I/O and CPU processing requires a high disk I/O bandwidth and a fast CPU. Sequential I/O is a very fast I/O pattern. Performance benefits from preserving the sequential characteristics on a disk. So we did not mix this pattern with any other disk I/O load on the same disk.

This import process itself can use only one CPU, but there is no specific affinity to a certain CPU - therefore we cannot conclude from the performance data, how much load on a certain CPU is caused by the import process and how much from the database. The average CPU utilization was about 64% and 5% I/O wait for one CPU, and about 49% with 3% I/O wait for the other. The total CPU utilization was of about

110% (where 100% is one CPU), indicating that there is also significant effort from the database (see bullet 2.). The value of 5% I/O wait can be considered as low, reading the import data is one reason for this value.

2. The database waits to get the input from the import process and then inserts it into the tables. After this it waits again for new data from the import process. These activities are very short and switch with a high frequency, so they cannot be isolated in the performance data. From the performance perspective, the database has two types of disk I/O load patterns:

- a. Inserting new data and creating indices is randomized read and write I/O (DBW0)
- b. Writing the logs is pure sequential write (LOGWR)

As before, the sequential write pattern of the log related disk I/O should be preserved instead of being mixed with any other activity on the same disk. Writing the log is also a candidate to cause I/O wait. Our low value of I/O wait indicates that the setup and the disk subsystem provide sufficient I/O bandwidth.

The database management system provides a very efficient large buffering system and many optimizations for the data related disk I/O, especially when using Direct I/O and Asynchronous I/O (`filesystemio_options = setall`).

3. Once a table is loaded, the corresponding indices are created. This is a very special phase. Now the import process is waiting, while the database management system shows all its strengths in dispatching and balancing workloads. This phase works with multiple processes, the more CPUs are available, the shorter this phase. During this phase both CPUs are fully utilized and there is nearly no I/O wait, but an impressive read rate of about 250 MB/sec.

In summary, the performance of the import process determines the behavior of the whole system - it needs a fast CPU and a high disk I/O bandwidth. For the database logs, focusing on separate disks and a high disk I/O bandwidth is also important. For the processing of the imported data, we rely on the disk I/O optimization features of the database management system for the data I/O stream itself. The index creating phase benefits from fast and from multiple CPUs. We expected the whole process to improve significantly when migrating from System z9 to System z10, just because of the faster CPUs.

The import process runs locally on the same system as the database management system, so we do not need to consider network performance.

### Architecture comparison

We imported an export file of 20 GB containing data from multiple table spaces into an Oracle database with explicit index creation. We used a z/VM guest with 2 CPUs and 16 GiB memory (see Table 1) on a System z9 and on a System z10 and measured the required time.

The import was initiated using the following statement

```
imp dbuser/passwd PARFILE=PARFILE.imp FILE=/loaddata/loadtest.dmp  
with the following import configuration file (PARFILE.imp):
```

```
BUFFER=5242880  
SHOW=N  
IGNORE=Y  
GRANTS=Y  
INDEXES=Y  
COMMIT=Y  
CONSTRAINTS=Y  
FROMUSER=(dbuser)  
TOUSER=(dbuser)
```

The resulting runtimes are shown in Figure 3, normalized that the z9 runtimes are 100%.

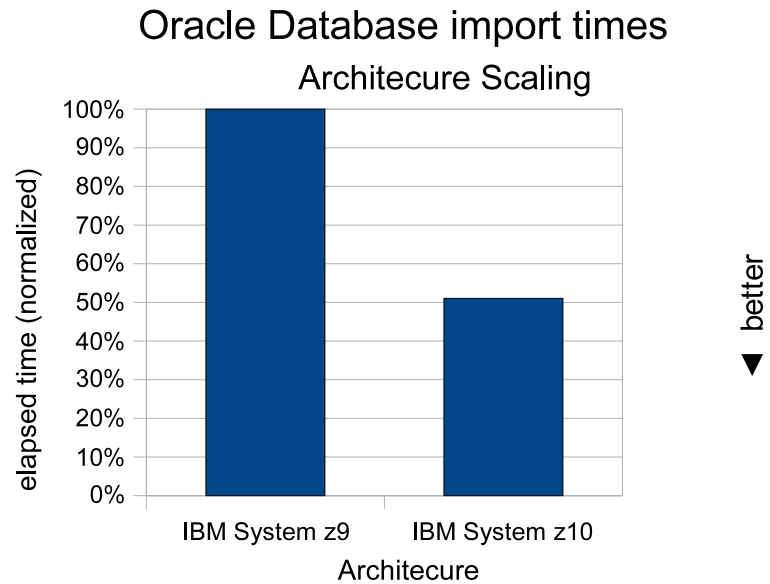


Figure 3. Execution times required for importing 20 GB data and index creation on System z9 and System z10.

The chart shows that this process needs only half the time on System z10 than on System z9, which is an excellent improvement for this scenario.

During the import phase, we observed a disk I/O rate of about 70 MB/sec (read and write), which can be considered as a high load for database I/O. During index creation, the disk I/O rate was about 250 MB/sec (read), which is a really heavy load. Both values show the strengths of the I/O setup with the separation of the I/O patterns and the striped storage pools on the DS8000.

### Summary

In this specific customer setup of importing data into an Oracle 10g R2 database running Red Hat Linux under z/VM 5.3 on a System z, we could show an improvement of a factor of two when migrating from System z9 to System z10.<sup>2</sup> The factor of two is based on the compute intensive workload which was neither burdened by heavy network load nor by the storage server which handled the data rates of up to 250 MB/sec easily.

The import process itself may be further improved by using the new Oracle Data Pump utility or by creating multiple export files - for example one per table space, and importing them in parallel. Another alternative might be to use ASM to manage the database files.

This very impressive result helps confirm that upgrading to the IBM System z10 can significantly enhance the capabilities of enterprise IT customers.

<sup>2</sup>Performance is 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.





Copyright IBM Corporation 2009  
IBM Systems and Technology Group  
Route 100  
Somers, New York 10589  
U.S.A.  
Produced in the United States of America,  
05/2009  
All Rights Reserved

IBM, IBM (logo), DS8000, System z, System z9, System z10, z9, z10 EC, and z/VM are trademarks or registered trademarks of the International Business Machines Corporation.

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.

InfiniBand and InfiniBand Trade Association are registered trademarks of the InfiniBand Trade Association. 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 statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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.