Linux on IBM zSystems and LinuxONE

Improving Transactional Database Workloads in KVM Guests on IBM zSystems using Huge Pages

About this publication

Before using this information and the product it supports, please read the information in section "7 Notices and disclaimer" on page 13.

This study describes how huge pages can improve a setup with KVM where the guest runs a transactional database workload. It also gives advice on how to set up huge pages and what needs to be considered.

Author: Dr. Juergen Doelle

Table of Contents

1 Introduction

Computer architectures structure the memory space in pages to manage it. Historically, the page size is 4 KiB. Newer computer architectures meanwhile support much larger page sizes like 1 MiB or even larger, these are called huge pages. Using huge pages reduces the number of pages needed for the same memory size significantly because the single page is much larger than the default page with 4KiB. This causes a severe reduction in effort for the operating system since the lists for managing memory pages shrink significantly due to the lower number of pages to address. This can be for example monitored with the size of the page tables.

Thus, it can be a preferable setup option as the operating system can manage huge pages in a more efficient way, especially for larger systems. Huge pages can be easily configured and only need support from the underlying hardware architecture. IBM zSystems® and LinuxONE have supported huge pages for years.

This study on an IBM® z15® shows the impact of huge pages in our environment with one KVM guest running a transactional database workload.

2 Summary

There are two possibilities where huge pages can be used in a KVM environment:

- Guest memory:
  In the host, the guest memory can be backed with huge pages. This is transparent to the guest as the guest only sees one address space without notifying the underlying structures.
- Application memory:
  Inside the guest, applications using large buffers often support the usage of huge pages for these buffers. In case multiple Linux processes sharing large buffers an additional effect comes up, as each process needs to map each memory page accessed into its address space. Thus, the space needed for page tables grow further.

All combinations of these two setup possibilities are considered in this study. Backing the guest memory with huge pages results in a reduction of CPU cycle consumption and additionally significantly reduces host memory needed for page tables. When using huge pages for the database buffer, the CPU utilization decreases as well. Furthermore, significant reduction of page faults and memory needed for page tables can be achieved in the guest.

Combining both setups in the test environment
- backing the guest memory with huge pages and
- using huge pages for database buffers,

results in a reduction in CPU utilization of more than one CPU without considerable impact on throughput rates. These effects would scale with the number of guests and the size of the relevant memory areas like guest size and buffer size. That means the larger the guests and the higher the number of guests, the more the usage of huge pages can help to reduce the CPU utilization of such an environment.

It should be mentioned that the usage of huge pages has some impacts needing to be considered:

- Huge pages cannot be swapped out. This part of the memory cannot be over-committed.

- It requires planning of the memory requirements. It is not possible to use a mix of huge and 4K pages for a specific buffer. For example, if not all required huge pages are available, the guest will fail to start. Another issue which might happen here is, if a database cannot place the whole buffers into huge pages, it might place them in the 4K memory causing an unintended memory pressure there.
  Therefore, it is good practice to reserve some more pages than intended and verify at the beginning how many pages are allocated.

In case memory over-commitment in the host is needed, an option might be to use only huge pages in the guest. The guest can still not swap out huge pages and needs to be sized appropriately. But when the host backs the guest memory with 4 KiB pages, the guest memory can be swapped out to the swap area in case of a memory shortage in the host. However, that will result in a performance impact on the application waiting for pages to get swapped in again. This should be used carefully.

3 Environment

The setup of the test environment's individual components, like host, guest, disk, workloads, and database, is described in this section.

3.1 Host

The test environment runs on an LPAR on IBM z15:

- Number of cores: 8
- Number of threads/CPUs: 16
- Memory: 200 GiB
- FICON Express16SA: 8x

Operating system:

- RHEL 8.6 - AV
- Kernel linux-4.18.0-372.9.1.el8.s390x
- qemu-kvm-6.2.0-11
- libvirt-8.0.0-5

To support guest configurations with huge page memory backing, the kvm module must be loaded with the parameter hpage=1. This can be done by specifying kvm.hpage=1 on the kernel parameter line. For more information see the *Device Drivers, Features, and Commands* book at https://www.ibm.com/docs/en/linux-on-systems?topic=configuration-device-drivers-features-commands.

3.2 KVM Guest Setup

The KVM guest is set up as follows:

- Number of virtual CPUs: 8
- Memory: 64 GiB

- Disk access via virtio, disk type="block" device="disk", cache="none" io="native"
- The default iothread setup was used

Operating system:

- RHEL 7.9
- Kernel linux-3.10.0-1160.el7.s390x

3.3 DASD Setup

ECKD devices from a storage server IBM DS8950F Model 996 were used:

- Disks from 4 LCUs
- Each LCU provides
  - 1x data disk 256 GiB (mod335)
  - 1x log disks 20.63 GiB (mod27)
  - 11x HyperPAV alias devices

This results in a total of 1 TiB raw disk space for data and 82.5 GiB for database logs.

3.4 Database

A transactional database was used.

- The initially loaded database size was about 300 GiB data.
- The database buffer size was about 20 GiB.

3.5 Workload

The workload was a transactional database workload using warehouse-related transactions. The number of users executing transactions was varied with 100, 250 and 500 to scale the workload levels and memory requirements.

The workload generator runs the workload locally in the same guests to avoid network effects.

All combinations of two setup variations were tested:

- Backing the guest pages in the host with huge pages.
- Providing huge pages for the database buffers.

Each measurement point was executed three times. The values shown in this paper are averages from these three runs.

4 How to set up huge pages

This section describes how to reserve huge pages via libhugetlbfs using sysctl.conf. The automated application of this setting at boot time requires that the corresponding service systemd-sysctl.service is enabled in host and guest. For the guest further settings are required in the domain.xml file and a specific kernel parameter.

A huge page size of 1 MiB was used, because the utilized version of qemu does not support 2 GiB huge pages.

4.1 Host

In the host, huge pages are reserved for the guest and the domain.xml was modified:

- For the guest with 64 GiB memory, huge pages are reserved using `vm.nr_hugepages=70000` in sysctl.conf. Another possibility could be to allocate the appropriate number of huge pages via `virsh allocpages 1M 70000` before starting the guest.

- For backing guest memory in the host via huge pages define
  ```
  <memoryBacking>
  <hugepages/>
  </memoryBacking>
  ```
  in the guest's domain.xml (see also https://libvirt.org/formatdomain.html#memory-backing).

To verify the usage of huge pages check /proc/meminfo in the host:

```
HugePages_Total:   70000
HugePages_Free:    70000
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       1024 kB
```

This output shows the state before the guest was started:

- The number of reserved huge pages: `HugePages_Total`
- The number of free huge pages: `HugePages_Free`

Here both values are identical, meaning the huge pages are reserved but not allocated.

After starting the KVM guest backed with huge pages, this changes to the following:

```
HugePages_Total:   70000
HugePages_Free:     4464
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       1024 kB
```

The difference between HugePages_Total and HugePages_Free shows the number of huge pages allocated from the host for the guest memory. In the example above the number of reserved pages might be reduced according to the relatively high number of free huge pages.

To support guest configurations with huge page memory backing, the kvm module must be loaded with the parameter hpage=1. This can be done by specifying kvm.hpage=1 on the kernel parameter line. For more information see the *Device Drivers, Features, and Commands* book at https://www.ibm.com/docs/en/linux-on-systems?topic=configuration-device-drivers-features-commands.

For KVM guests running in an OpenStack environment, more information can be found at https://www.ibm.com/support/pages/backing-your-guests-hugepages.

4.2 Guest

To provide huge pages in the guest for the database, reserve an appropriate number of the huge pages via libhugetlbfs (for example via `vm.nr_hugepages=21000` in sysctl.conf for a 20 GiB buffer).

The reservation and usage of huge pages can be verified in the guest via /proc/meminfo, the same way as in the host.

It is important that sufficient space is available in the huge page area. The buffer of interest must fit completely, otherwise the application might fail or allocate the buffer in the 4K area, which might cause unintended memory pressure there.

5 Results

All combinations of the setup variants were tested:

Host setup: Backing guest memory	Guest setup: Database uses	Abbreviation
4K pages	4K pages	4K Backing/4k Guest
huge pages	4K pages	HP Backing/4k Guest
4K pages	huge pages	4K Backing/HP Guest
huge pages	huge pages	HP Backing/HP Guest

There is no contention for CPU or memory usage in all scenarios.

5.1 Performance Metrics

The most important performance metrics are throughput and CPU load. This section describes how the usage of huge pages impacts these parameters, especially how it influences the CPU cost of the throughput.

5.1.1 Throughput

The following chart shows the transactional throughput in all four scenarios.

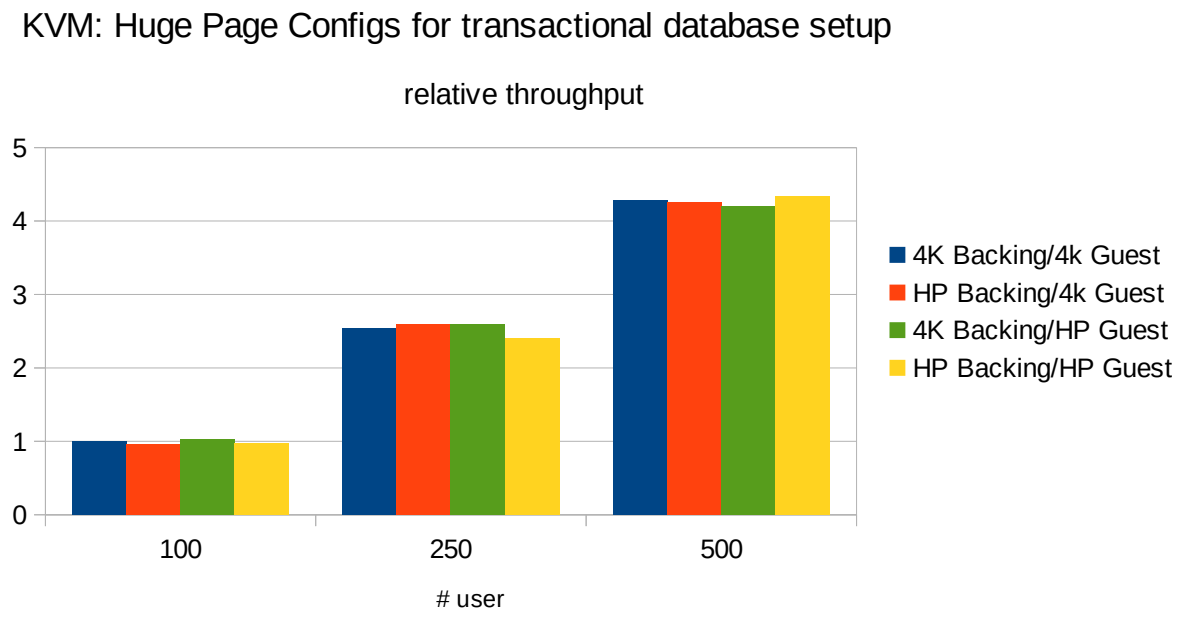

Figure 1: Transactional throughput for scaling huge page setups

As Figure 1 shows, the throughput in our environment is similar in all scenarios.

5.1.2 CPU load on the host

Figure 2 shows the total CPU load in the host in units of CPUs. This includes the load from the guest.
Note: The guest has eight virtual CPUs, the host has 16 CPUs.

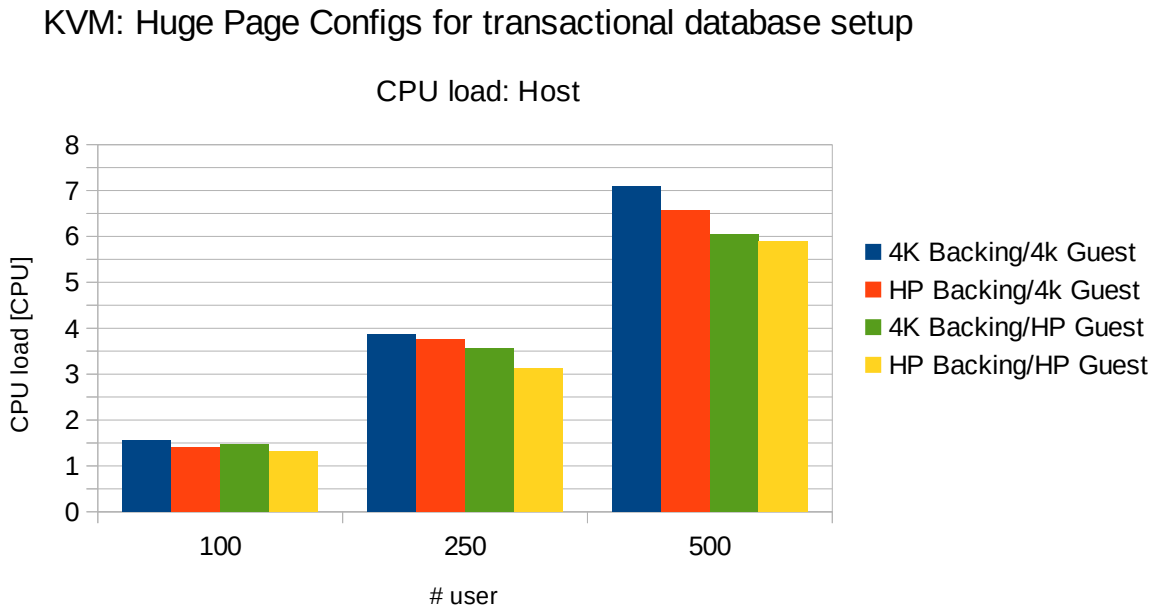

Figure 2: CPU load in the host for scaling huge page setups

- Guest memory: Enabling huge pages for backing the guest memory reduces CPU utilization.
- Application memory: Enabling huge pages in the guest for the database buffers reduces CPU utilization.

The largest difference to 4K pages for guest backing and database buffers appears at the highest load level with 500 users. Here, the usage of huge pages for both reduces the CPU load from more than seven CPUs to less than six CPUs for the same workload, resulting in a reduction of 15%.

5.1.3 Throughput per CPU

This metric shows how 'expensive' the throughput is in terms of how many transactions can be driven with a certain amount of CPU cycles. This mitigates effects from variations in CPU load due to throughput variations.

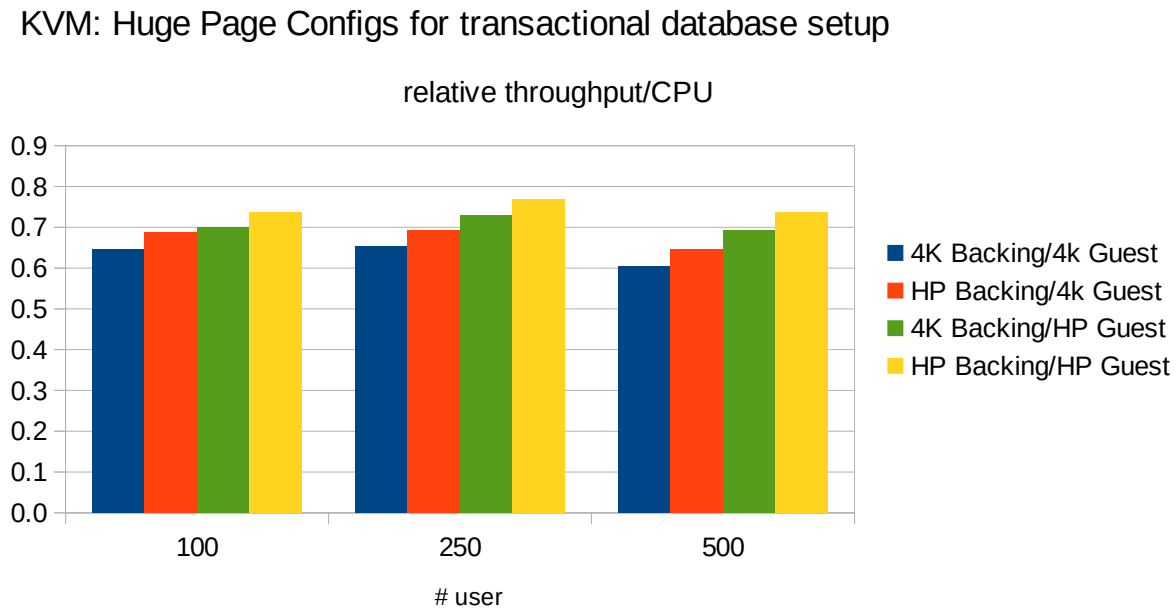

Figure 3: Throughput driven per CPU for scaling huge page setups

Figure 3 shows that both configurations, enabling huge pages for the guest memory and enabling huge pages for the application, contribute to the total improvement.

5.2 Improvements from huge pages to other system parameters

Using huge pages also has an impact on less obvious parameters related to Linux internal system management. These indicate the source of the CPU savings.

5.2.1 Size of page tables

The amount of memory needed for managing the guest pages is an important parameter for memory requirements in the host and effort for accessing the guest memory.

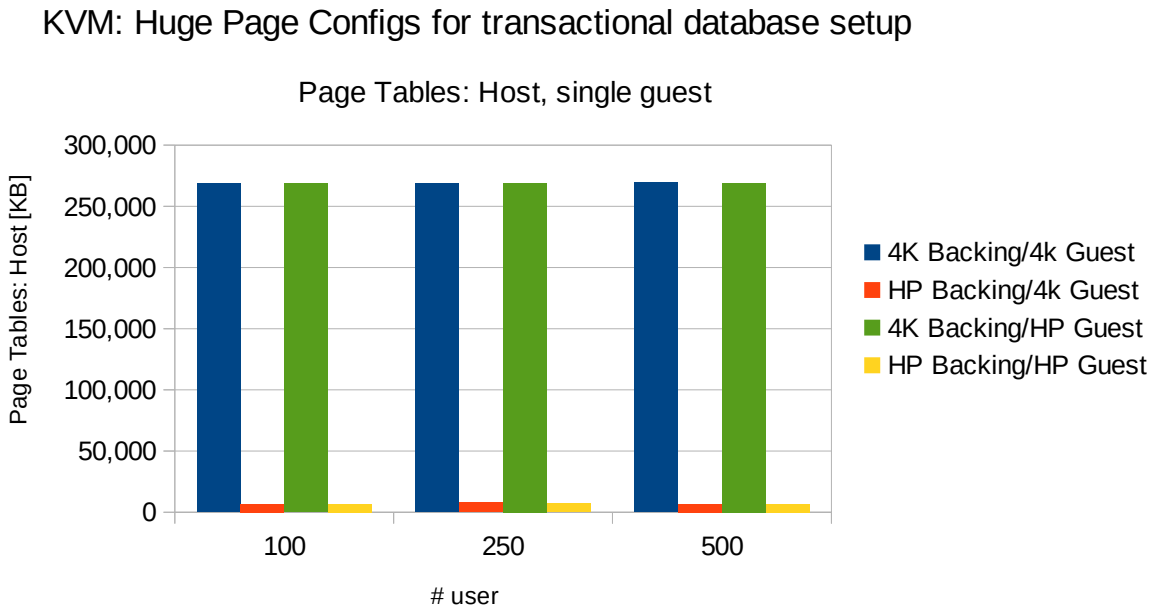

Figure 4: Size of page tables in the host for scaling huge page setups

Figure 4 shows that the size of the page tables in the host shrink for a single guest with 64 GiByte memory size from ca 260 MiB to 7 MiB. This saving would scale with the number of guests and the size of the guest size. It also shows that this parameter is independent from the workload level. This is expected because it just reflects the number of accessed pages but not how intensively they are used.

This effect appears in larger dimensions in the guest. When using huge pages for the database buffers, the test system needed only 1.5% of the page table memory as compared to a setup with 4K pages. This effect scales with the buffer size and with the number of guests in such an environment.

5.2.2 Number of page faults

The number of page faults is a parameter which influences the effort and time needed to access pages. A high number indicates more CPU cycles and longer response times for accessing a page than a lower number.

Figure 5 shows the impact of using huge pages for the database buffer in the guest.

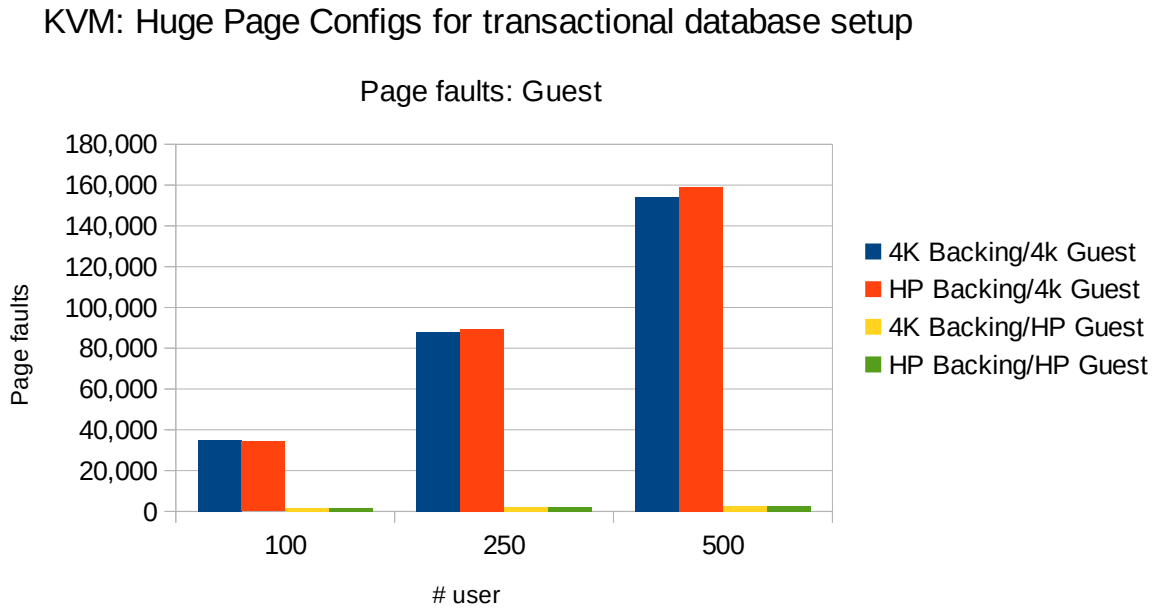

Figure 5: Page faults in the guest when scaling huge page setups

The page faults in the guest can be drastically reduced when the database uses huge pages for its buffers, which in turn leads to the reduction of CPU cycles needed to access the memory buffers. This scales with the buffer size.

6 Conclusion

Combining both setups, backing the guest memory with huge pages and using huge pages for database buffers, results in a reduction in CPU utilization of more than one CPU without considerable impact on throughput rates in our test environment. The shrinking memory requirements for page tables in host and guest and the reduced number of page faults in the guest are the cause for these effects.
This scales with the number of guests and the size of the relevant memory areas, like guest size and buffer size. That means the larger the guests and the higher the number of guests, the more the usage of huge pages can help to reduce CPU utilization of such an environment.

7 Notices and disclaimer

U.S. Government Users Restricted Rights – use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.

Any statements regarding IBM’s future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environment. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer’s responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer’s business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of 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. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM’s products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

Trademarks

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right. Refer to www.ibm.com/legal for further legal information.