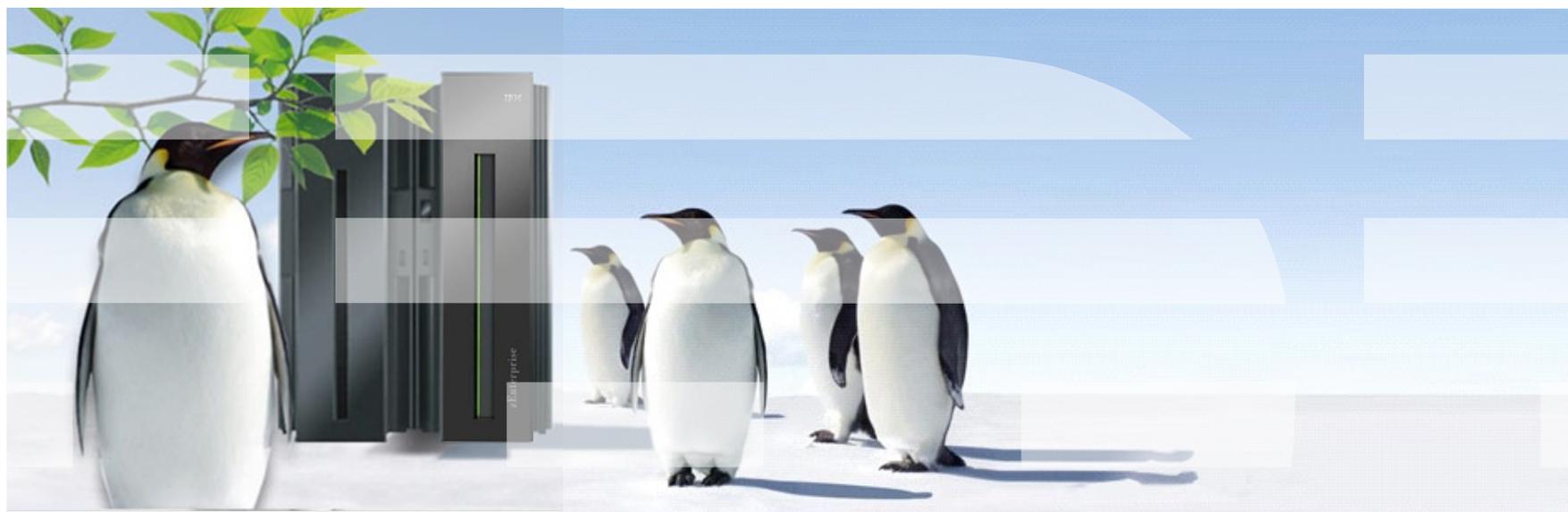


What's New in Linux on IBM z Systems

Martin Schwidefsky
IBM Lab Böblingen, Germany
April 15, 2015



Trademarks & Disclaimer

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

IBM, the IBM logo, BladeCenter, Calibrated Vectored Cooling, ClusterProven, Cool Blue, POWER, PowerExecutive, Predictive Failure Analysis, ServerProven, System p, System Storage, System x , z Systems, WebSphere, DB2 and Tivoli are trademarks of IBM Corporation in the United States and/or other countries. For a list of additional IBM trademarks, please see <http://www.ibm.com/legal/copytrade.shtml>.

The following are trademarks or registered trademarks of other companies: Java and all Java based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries or both Microsoft, Windows, Windows NT and the Windows logo are registered 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 or both. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc. InfiniBand is a trademark of the InfiniBand Trade Association.

Other company, product, or service names may be trademarks or service marks of others.

NOTES: Linux penguin image courtesy of Larry Ewing (lewing@isc.tamu.edu) and The GIMP

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Users of this document should verify the applicable data for their specific environment. IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Information is provided "AS IS" without warranty of any kind. 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 are suggested US list prices and are subject to change without notice. Starting price may not include a hard drive, operating system or other features. Contact your IBM representative or Business Partner for the most current pricing in your geography. Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use. The information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any

Notice Regarding Specialty Engines

Any information contained in this document regarding Specialty Engines (“SEs”) and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”).

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Linux on IBM z Systems introduction

Interesting facts and numbers

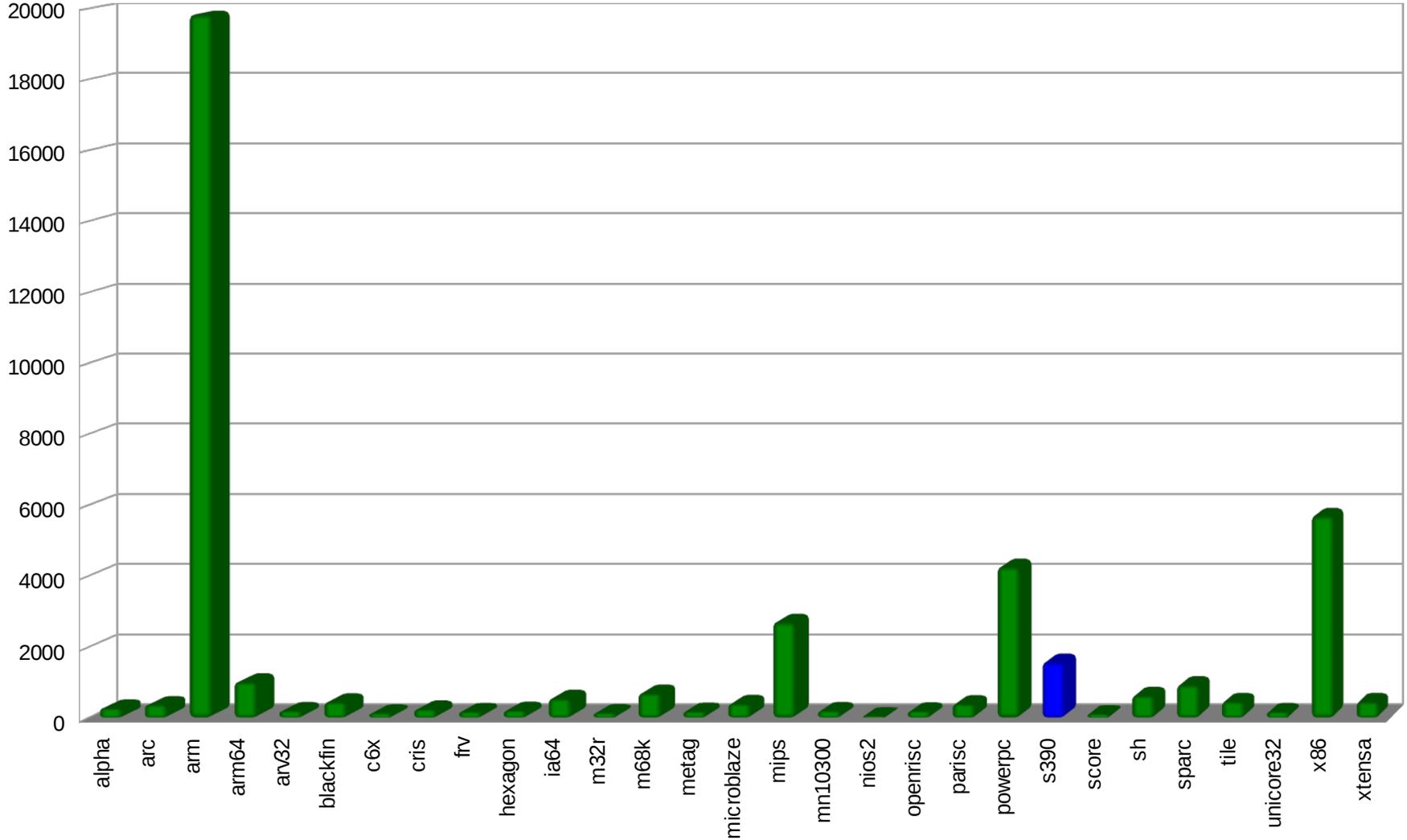
Fun facts around Linux on IBM z Systems

- Linux kernel v3.19 has 19,129,942 lines of code
How many lines of code had the initial patch for s390 ?
36,811 lines of code (patch size)
- How many lines of code are specific to s390 in v3.19 ?
213,285 lines of code (1.11%)
- How many lines of assembler code are in v3.19 ?
396,473 lines in 1376 files, 4653 specific to s390 (1.17%)
- How many individual developers contributes to Linux in 2014 ?
3697 individuals, 1225 with a single commit, 32 contributed directly to s390
- What is the architecture with the larges amount of core changes ?
ARM, with ~105 KLOC per release for v3.x, followed by mips ~24 KLOC, powerpc ~23 KLOC, and x86 ~22 KLOC. s390 has an average of 7 KLOC.

Linux is Linux, but ...features, properties and quality differ dependent on your platform and your use case

Source: <http://kernel.org>
<http://linuxcounter.net>

git commits per architecture in 3.x

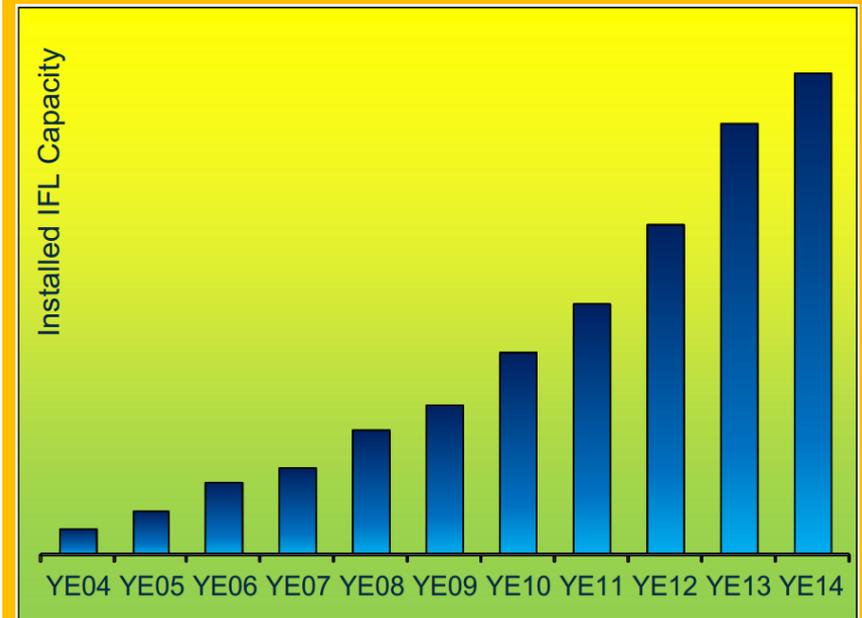


Linux on IBM z Systems in 4Q2014

*Installed Linux MIPS at 45% CAGR**

- 27.3% of Total installed MIPS run Linux as of 4Q14
- Installed IFL MIPS increased 12% from 4Q13 to 4Q14
- 39% of z Systems clients have IFL's installed as of 4Q14
- 82 of the top 100 z Systems clients are running Linux on the mainframe as of 4Q14 **
- 35% of all z Systems servers have IFLs
- 60% of new FIE/FIC z Systems accounts run Linux

Installed Capacity Over Time



Linux on IBM z Systems distributions

What is available today

Linux on IBM z Systems distributions in service

- **SUSE Linux Enterprise Server 10 (GA 07/2006)**
 - Kernel 2.6.16, GCC 4.1.0, Service Pack 4 (GA 04/2011)
- **SUSE Linux Enterprise Server 11 (GA 03/2009)**
 - Kernel 2.6.27, GCC 4.3.3, Service Pack 1 (GA 06/2010), Kernel 2.6.32
 - Kernel 3.0, GCC 4.3.4, Service Pack 3 (GA 07/2013)
- **SUSE Linux Enterprise Server 12 (GA 10/2014)**
 - Kernel 3.12, GCC 4.8
- **Red Hat Enterprise Linux AS 5 (GA 03/2007)**
 - Kernel 2.6.18, GCC 4.1.0, Update 10 (GA 10/2013)
- **Red Hat Enterprise Linux AS 6 (GA 11/2010)**
 - Kernel 2.6.32, GCC 4.4.0 Update 5 (GA 11/2013)
- **Red Hat Enterprise Linux AS 7 (GA 06/2014)**
 - Kernel 3.10, GCC 4.8, Update 1 (GA 03/2015)
- **Others**
 - Debian, Slackware,
 - Support may be available by some third party

Supported Linux Distributions

Distribution	z13	zEnterprise - zBC12 and zEC12	zEnterprise - z114 and z196	System z10 and System z9
RHEL 7	✓ (1,3)	✓ (4)	✓ (4)	✗
RHEL 6	✓ (1,3)	✓ (5)	✓	✓
RHEL 5	✓ (1,3)	✓ (6)	✓	✓
RHEL 4 (*)	✗	✗	✓ (9)	✓
SLES 12	✓ (2,3)	✓	✓	✗
SLES 11	✓ (2,3)	✓ (7)	✓	✓
SLES 10 (*)	✗	✓ (8)	✓	✓
SLES 9 (*)	✗	✗	✓ (10)	✓

- ✓ Indicates that the distribution (version) has been tested by IBM on the hardware platform, will run on the system, and is an IBM supported environment. Updates or service packs applied to the distribution are also supported.

Please check with your service provider which kernel-levels are currently in support.

See www.ibm.com/systems/z/os/linux/resources/testedplatforms.html for latest updates and details.

Current Linux on IBM z Systems Technology

Key features & functionality already
contained in the SUSE & Red Hat Distributions

IBM zEnterprise EC12 and BC12 support

▪ Transactional execution (kernel 3.7)

- Also known as hardware transactional memory
- CPU features that allows to execute a group of instructions atomically
- Optimistic execution, if a transaction conflicts a rollback to a saved state is done



▪ Storage class memory – Flash Express (kernel 3.7)

- Internal Flash Solid State Disk (SSD)
- Accessed via Extended Asynchronous Data Mover (EADM) sub-channels
- Support for concurrent MCL updates with kernel version 3.8



▪ Support for Crypto Express 4S cards (kernel 3.7)

- New generation of crypto adapters plug-able into the I/O drawer
- New type 10 which uses a bit field to indicate capabilities of the crypto card



▪ Native PCI feature cards (base in kernel 3.8, ongoing)

- Support for native PCIe adapters visible to the operating system



▪ Oprofile zEC12 hardware sampling support (kernel 3.10)

- Extend the hardware sampling to support zEC12



System zEC12 features – Transactional Execution



- **Transactional execution is a concurrency mechanism of the CPU comparable to database transactions**
 - Several reads and stores from/to memory logically occur at the same time
 - Improves performance for fine-grained serialization
 - Useful for lock-less data structures and speculative compiler optimizations
- **Two types of transactions: constraint and non-constraint**
- **Conflicting memory accesses will cause the transaction to abort**
 - Transaction abort is rather expensive
 - Constraint transaction will automatically restart
 - Ratio of successful vs. aborted transaction is important for performance
- **Kernel support is required to enable user programs to use transactional execution**
 - Control registers setup
 - Debugging support for additional PER controls via ptrace

System zEC12 features – Transactional Execution



11.3



6.4

Example of a list_add operation

```

struct spinlock_t list_lock;
struct list_head list_head;
void list_add(struct list_head *new)
{
    spin_lock(&list_lock, 0, 1);
    list_add(new, &list_head);
    spin_unlock(&list_lock, 1, 0);
}

```

Typical pattern:
1) lock, 2) a short operation, 3) unlock

Traditional code:

```

# spin_lock
    larl    %r3,list_lock
    lhi     %r1,1
lock:  lhi     %r0,0
       cs     %r0,%r1,0(%r3)
       ltr    %r0,%r0
       jne    lock
# list_add
    larl    %r4,list_head
    lg     %r5,0(%r4)
    stg    %r4,0(%r2)
    stg    %r5,8(%r2)
    stg    %r2,0(%r5)
    stg    %r2,8(%r4)
# spin_unlock
    cs     %r1,%r0,0(%r3)
    br     %r14

```

Transactional code

```

# begin transaction
tbegin 0,0

# list_add
    larl    %r4,list_head
    lg     %r5,0(%r4)
    stg    %r4,0(%r2)
    stg    %r5,8(%r2)
    stg    %r2,0(%r5)
    stg    %r2,8(%r4)
# end transaction
tend

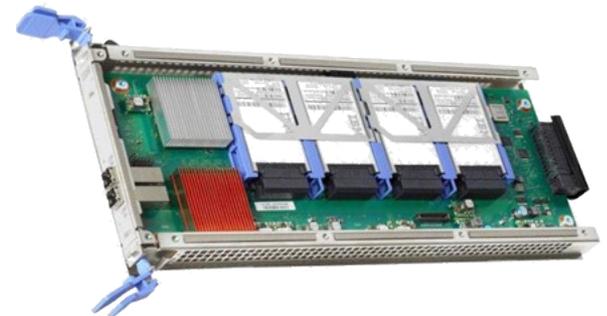
br     %r14

```

zEC12/zBC12 features – Flash Express

 11.3 6.4

- **PCIe I/O adapter with NAND Flash SSDs**
 - Flash Express cards are plugged as pairs to build a RAID10
 - Pair is connected with interconnect cables
 - Card replacement is concurrent if one card fails
 - Up to 4 pairs of cards are supported (4 * 1.4TB = 5.6TB)
- **New tier of memory: Storage Class Memory**
 - Accessed via Extended Asynchronous Data Mover (EADM) subchannels via the new Storage Class Memory (SCM) block driver
- **Flash Express is split into memory increments**
 - Memory increments are assigned to LPARs via the SE or HMC
 - Memory increment size is 16 GB
- **Flash Express is not persistent over IML**



PCI support



- **Native PCIe feature cards introduced on zEC12 and zBC12**
 - 10GbE RoCE Express, network card for SMC-R
 - zEDC Express, data compression/decompression card

- **Native PCIe adapter concept**
 - Plugged into an PCIe I/O drawer
 - Managed by an internal firmware processor (IFP)
 - Device driver for the PCIe function is located in the operating system

- **Uses standard Linux PCI support and drivers with some constraints**
 - Only MSIX, no port I/O, memory mapped I/O by use of PCI load/store instructions
 - Provides ability to assign individual functions of an adapter to an LPAR
 - Converted z Systems architecture code to use generic hardirqs
 - Only selected PCIe adapters are known to the IFP and surfaced to the OS

10GbE RoCE Express



■ Native PCIe networking card

- 10 Gigabit remote direct memory access (RDMA) capable network card
- Uses Infiniband RDMA over Converged Ethernet (RoCE) specification
- Up to 16 10GbE RoCE Express adapters per machine
- Reduced latency and lower CPU overhead
- Supports point-to-point connections and switch connection with an enterprise-class 10 GbE switch

■ Software support

- z/OS V2R1 with PTFs supports SMC-R with RoCE
- z/VM support planned
- Linux support in available upstream and as tech preview in SLES12 / RHEL7



zEDC Express



- **Native PCIe data compression / decompression card**
 - Up to 8 adapters can be installed into a single machine
 - With large blocks, it can compress data at more than 1 GB per second
 - Implements compression as defined by RFC1951 (DEFLATE)
 - Comparable to “gzip -1”
- **Software support**
 - z/OS V2R1, V1R13 and V1R12 with PTFs
 - Linux device driver to gain access to zEDC has been posted on LKML and has been accepted into the upstream kernel
 - The zlib open source library is a C implementation commonly used to provide compression and decompression services.



Linux on IBM z Systems features – Compiler toolchain

▪ zEnterprise 196 exploitation (gcc 4.6)



- Use option `-march=z196` to utilize the new instructions added with z196
- Use `-mtune=z196` to schedule the instruction appropriate for the new out-of-order pipeline of z196
- Re-compiled code/apps get further performance gains through 110+ new instructions

▪ zEC12/zBC12 exploitation CPU (gcc 4.8)



- Use option `-march=zEC12` to utilize the instructions added with zEC12
- Use option `-mtune=zEC12` to schedule the instructions appropriate for the pipeline of zEC12
- zEC12/zBC12 comes with new instructions
 - Transactional Memory support
 - Improved branch instructions

Emerging Linux on IBM z Systems Technology

Upcoming features & functionality from IBM

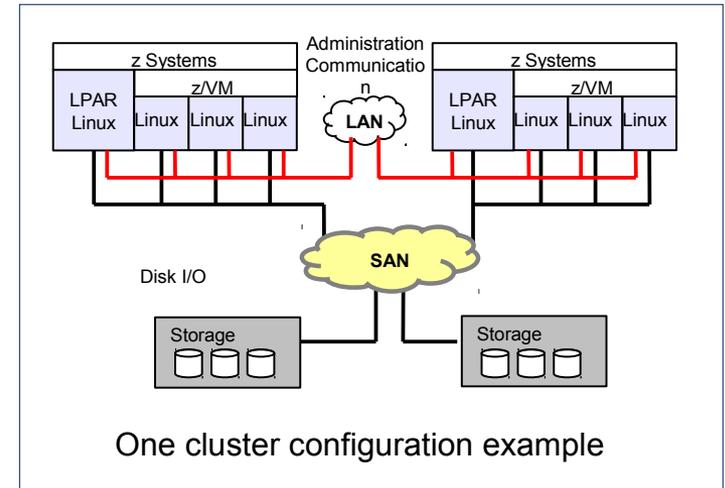
IBM Spectrum Scale for Linux on z Systems

Based on GPFS technology



Robust clustered file system

- Concurrent high-speed, reliable data access from multiple nodes
- Extreme scalability and accelerated performance
- Smooth, non disruptive capacity expansion and reduction



Linux instances in LPAR mode or on z/VM, on the same or different CECs

Up to 32 cluster nodes with same or mixed Linux distributions / releases

Support for ECKD™-based and FCP-based storage

Heterogeneous clusters w/ client nodes w/o local storage access running Linux on x86 or POWER®

Supported storage: DS8000®, IBM FlashSystem™ IBM Storwize® V7000, SVC, IBM XIV®,

Supported workloads: WebSphere App. Server, WebSphere MQ® or similar workloads

Support statements for first version

IBM XL C/C++ for Linux on z Systems, V1.1

High-performance compiler now delivered

■ XL C/C++ for Linux on z Systems offers:

- **Increased return on hardware investments** for improved application performance with leading-edge optimization technology and exploitation of the latest z Systems hardware.
- **The Clang infrastructure in the front end with advanced optimization technology** in the IBM compiler back end.
- **Highly tuned math libraries**, including Mathematical Acceleration Subsystem (MASS), Basic Linear Algebraic Subprograms (BLAS), and Automatically Tuned Linear Algebra Software (ATLAS).
- **Standards compliance** with support for international C and C++ programming language standards and GNU C/C++ compatibility extensions for ease of application migration to IBM z Systems.
- **Software requirements**
 - Red Hat Enterprise Linux for IBM System z 7
 - Red Hat Enterprise Linux for IBM System z 6.3
 - SUSE Linux Enterprise Server for System z 12
 - SUSE Linux Enterprise Server for System z 11 SP3

Available since Feb 2015

XL C/C++ for Linux on z Systems
60 day no-charge trial

Download software



More information:

• [Data sheet:](#)

IBM XL C/C++ for Linux on z Systems, V1R1



New, high-performance compiler delivered for Linux on z Systems

IBM® XL C/C++ for Linux on z Systems, V1.1 is a new XL C/C++ compiler for application development that takes advantage of the latest IBM z Systems servers that run on select Linux distributions. The XL C/C++ compilation technology for Linux on z Systems strengthens the platform, exploits the z Systems environment, and provides superior performance. A key strength of XL C/C++ for Linux on z Systems is its ability to generate highly optimized code for execution on IBM z Systems. It is the newest member of the IBM XL compiler family and is built on the performance gains from many years of IBM compiler optimization experience with existing XL C/C++ compilers that are available for IBM i/OS®, IBM z/VM®, IBM iSeries®, and Linux on IBM Power Systems®. With XL C/C++ for Linux on z Systems, you can create and port applications for execution on the next generation of IBM systems supporting select Linux distributions built on IBM z/Architecture™ technology designed for development of high-performing business applications and system programs, while maximizing hardware utilization with fast application performance.

Highlights

- XL C/C++ for Linux on z Systems, V1.1 delivers the following features:
 - Supports generation of highly optimized code exploiting z Systems servers.
 - Supports programming language standards, including partial support for the latest C11 and C++11 standards.
 - Provides a high level of source compatibility with GNU Compiler Collection (GCC) while providing binary coexistence.
 - Includes Automatically Tuned Linear Algebra Software (ATLAS) library.
 - Includes IBM Mathematical Acceleration Subsystem (MASS) library.
 - Includes Basic Linear Algebraic Subprograms (BLAS) functions.

Generation of highly optimized code

As the newest addition to the IBM compiler family, IBM XL C/C++ for Linux on z Systems, V1.1 brings mature IBM compiler experience to Linux distribution running on IBM z Systems servers. XL C/C++ supports generation of highly optimized code exploiting the z Systems servers, adheres to the latest programming standards, partially, and provides GNU Compiler Collection

(GCC) compatibility that allows you to easily port your applications to Linux distribution running on z Systems servers.

The Clang infrastructure

XL C/C++ for Linux on z Systems leverages the Clang infrastructure from the open source community for a portion of its compiler front end. Clang is a component of the LLVM open source compiler and toolchain project, and provides the C and C++ language front end for LLVM. XL C/C++ for Linux on z Systems combines the Clang front end infrastructure with the advanced optimization technology in the IBM compiler back end.

New architecture and tune compiler options for the z Systems technology

XL C/C++ for Linux on z Systems, V1.1 supports the new generation of z Systems hardware running SUSE Linux Enterprise Server 11 (SLES 11), SLES 12, Red Hat Enterprise Linux 6 (RHEL 6), and IBM z.

The `-qarch` compiler option specifies the processor architecture for which code is generated. The `-qopt` compiler option tunes instruction selection, scheduling, and other architecture-dependent performance enhancements to run best on a

ibm.com/software/products/en/czlinux

Future Linux on IBM z System Technology

Software which has already been developed
and integrated into the upstream packages

- but is **not yet available** in any
Enterprise Linux Distribution

IBM z13 support

- **Vector extension facility (kernel 3.18)**
 - Also known as single-instruction, multiple data (**SIMD**)
 - 32 128-bit vector registers are added to the CPU
 - 139 new instructions to operate on the vector registers
 - User space programs can use vectors to speed up all kinds of functions, e.g. string functions, crc checksums, ...
- **CPU multi threading support (> kernel 3.19)**
 - Also known as simultaneous multi-threading (**SMT**)
 - Once enabled the multi threading facility provides multiple CPUs for a single core.
 - The CPUs of a core share certain hardware resource such as execution units or caches
 - Avoid idle hardware resources, e.g. while waiting for memory
- **Extended number of AP domains (kernel 3.18)**
 - AP crypto domains in the range 0-255 will be detected
- **Crypto Express 5S cards (> kernel 3.19)**
 - New generation of crypto adapters with improved performance



IBM z13 – Vector extension, alias SIMD

▪ 32 vector registers with 128 bits each

- Register can be split into 16 bytes, 8 shorts, 4 integers, or 2 long integers
- Up to 4 concurrent 32x32 multiply / adds
- Many new vector instructions with many specialized use cases
 - e.g. Vector Galois Field Multiply Sum and Accumulate (VGFMA)

▪ Vector registers and floating pointer register partially overlap

- The program can use either the FPR or the VR with 0-15 at the same time

0	%f0 or %v0 bits 0:63	%v0 bits 64:127
...
15	%f0 or %v0 bits 0:63	%v0 bits 64:127
16	%v0 bits 0:127	
...	...	
31	%v0 bits 0:127	

IBM z13 – Vector extension, alias SIMD

▪ Vector instruction example: `size_t strlen(const char *s)`

```
# R2 address of the string, R1 will contain length
      XGR      R1,R1          Zero out running index
LOOP: VLBB     V16,0(R1,R2),6  Load up to 16 bytes
      LCBB     R3,0(R1,R2),6   Find how many bytes were loaded
      ALGRK    R1,R1,R3       Increment length by bytes loaded
      VFENEBZ  V17,V16,V16    Look for 0 byte
      VLVGB    R4,V17,7(R0)    Extract index to gpr (16-no match)
      CLGR     R3, R4         If GLEN <= GPOS have more to search
      BRNH     LOOP
      SLGRK    R1,R1,R3       Subtract off amount loaded
      ALGRK    R1,R1,R4       Add amount to the zero that was found
```

– 7 instructions to scan 16 bytes in the body of the `strlen` loop

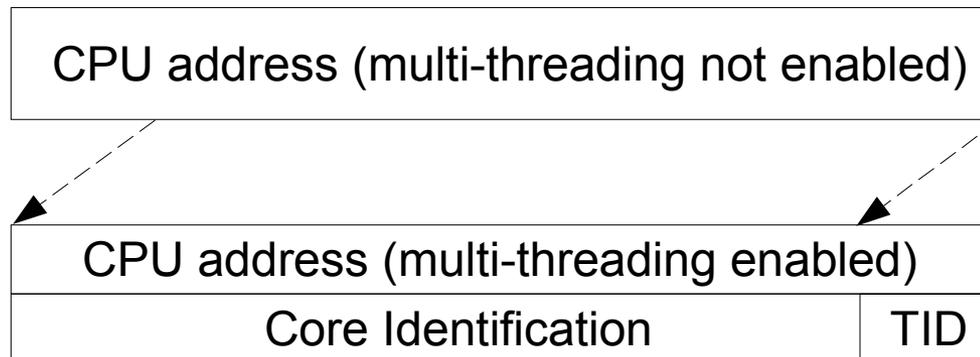
▪ Standard implementation uses loop unrolling for 8 bytes

– 1x 'CLI' + 1x 'BRC' for each byte, 1x 'LA' + 1x 'BRCT' for 8 bytes

→ 34 instructions to scan 16 bytes in the body of the `strlen` loop

IBM z13 – CPU multi threading (MT) support, alias SMT

- **Up to two hardware threads on a single core**
 - Known as simultaneous multi-threading (SMT) on z Systems and POWER, or Hyper-threading on x86
- **The operating system needs to opt-in to enable SMT**
 - The LPAR code starts a partition with one logical CPU per core
 - After the SIGP to enable MT additional logical CPUs are surfaced and the CPU addressing changes



- The additional CPUs work just like “normal” CPUs, but with different performance characteristics

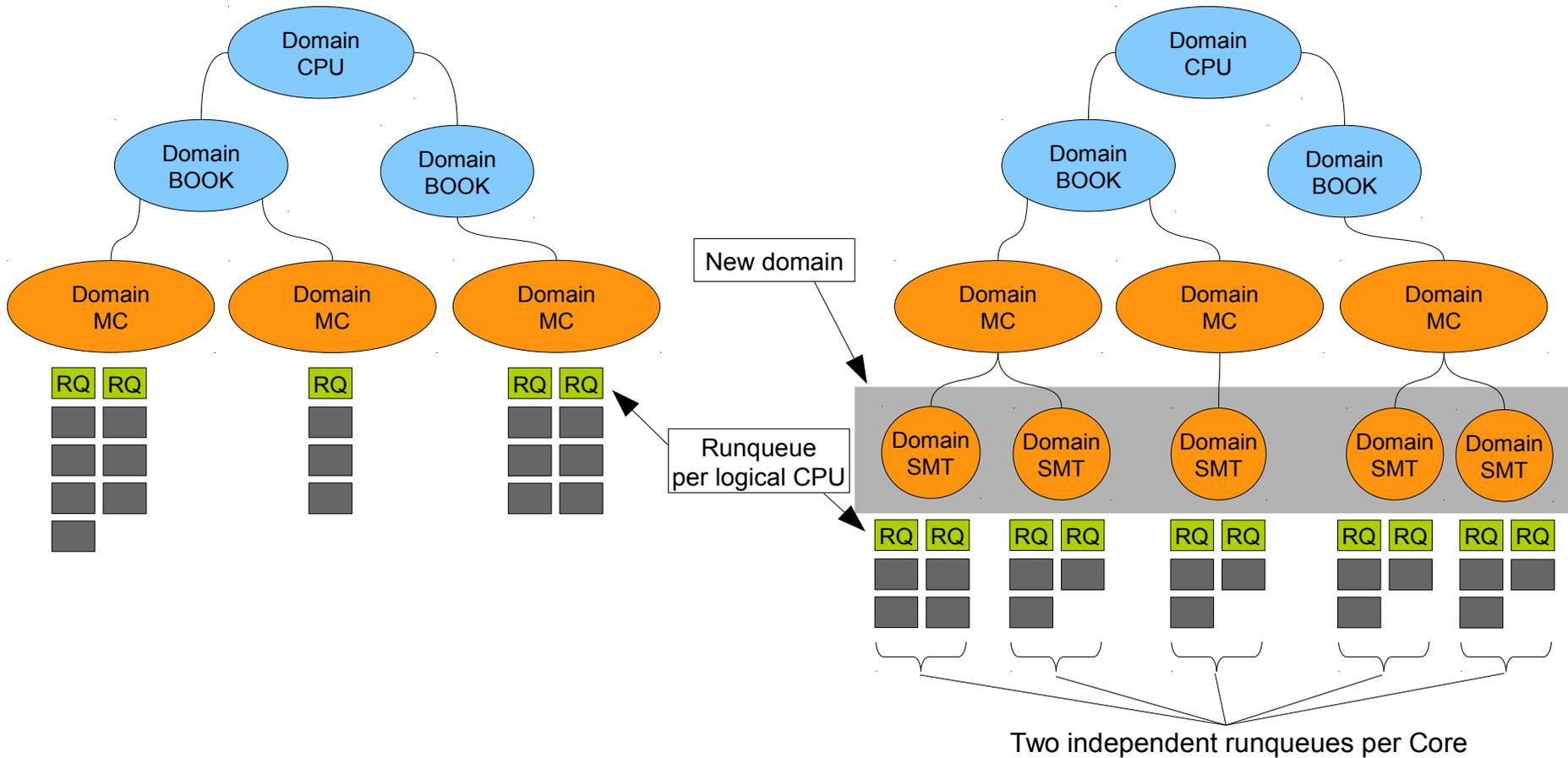
IBM z13 – CPU multi threading (MT) support, alias SMT

- Example lscpu output with MT enabled

```
# lscpu -e
CPU BOOK SOCKET CORE L1d:L1i:L2d:L2i ONLINE CONFIGURED POLARIZATION ADDRESS
0 0 0 0 0:0:0:0 yes yes horizontal 0
1 0 0 0 1:1:1:1 yes yes horizontal 1
2 0 1 1 2:2:2:2 yes yes horizontal 2
3 0 1 1 3:3:3:3 yes yes horizontal 3
4 0 1 2 4:4:4:4 yes yes horizontal 4
5 0 1 2 5:5:5:5 yes yes horizontal 5
6 0 1 3 6:6:6:6 yes yes horizontal 6
7 0 1 3 7:7:7:7 yes yes horizontal 7
8 0 1 4 8:8:8:8 yes yes horizontal 8
9 0 1 4 9:9:9:9 yes yes horizontal 9
```

IBM z13 – CPU multi threading (MT) support, alias SMT

▪ Scheduler CPU domains non-MT vs. MT



IBM z13 – CPU multi threading (MT) support, alias SMT

■ CPU time accounting with MT enabled

- New CPU-MF counters: number of cycles with n threads active
- Average thread density is calculated once per tick
- The raw CPU timer deltas are scaled with the average thread density
- The scaled CPU time is available via the 'taskstats' interface
- The standard CPU time values in /proc use the unscaled CPU time

sT_n scaled CPU time

T_n unscaled CPU time

C_i cycle count with i active threads

$$sT_n = T_n * \frac{\sum_{i=1}^n C_i}{\sum_{i=1}^n i * C_i}$$

IBM z13 – CPU multi threading (MT) support, alias SMT

- **Workloads may or may not benefit from MT**
 - Certain processor resource need to be underutilized for MT to be effective
- **Workloads characteristics with a positive effect on MT**
 - A large number of cache misses
 - Long chains of instruction dependencies
 - A large number of branch mis-predictions
- **Workload characteristics with a detrimental effect on MT**
 - A homogeneous instruction mix targeting a scarce function unit
 - Intensive use of the memory system with poor cache locality
 - Applications with poor scaling, in general MT requires more logical CPUs

IBM z Systems kernel features – memory management

■ **Implement fault based referenced page detection (kernel 3.12)**

- Convert referenced page detection from the reference-bit in the storage key to a fault based method. An old page is now always mapped with the invalid bit set (no read, no write access).
- New mappings are always created with the software referenced bit set
- Removes the storage key operations to detect page referenced state.

■ **Dirty bits for large segment table entries (kernel 3.17)**

- Allows to track the changed state of 1MB large pages
- Useful to avoid paging if a transparent-huge-page is split into 4K pages

■ **Avoiding storage key operations improves performance**

- The savings in storage key operations outweigh the slightly increase number of faults
- After IPL a system without KVM will not access the storage keys at all
- KVM still makes use of storage keys for provide correct guest virtualization

IBM z Systems kernel features – memory management

▪ **Convert bootmem to memblock (kernel 3.16)**

- The bootmem allocator needs one bits for each page during IPL
- With large memory size the bootmem bit arrays get huge, they are freed after IPL
- memblock uses lists of memory ranges which stay persistent

▪ **Uaccess rework (kernel 3.16)**

- The old uaccess code used a page table walk for some operations, e.g. the atomic futex operations
- The new solution is to temporarily switch from home space to primary space in the uaccess kernel code
- Avoid the costly page table walk and uses hardware DAT translation

IBM z Systems kernel features – core improvements

■ CPU-Measurement Sampling Facility (kernel 3.14)

- Uses the hardware CPU sampling facility to take snapshots of a set of sample data at a specified sampling interval, e.g. the cycle counter
- Integrated into the Linux 'perf' tool
- The basic-sampling mode and the diagnostic-sampling mode are supported
- The diagnostic-sampling mode is intended for use by IBM support only

■ Example how to record sampling data for an application

```
# perf record -e rB0000 - /bin/df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/dasda1 6967656 3360508 3230160 51% /
none 942956 88 942868 1% /dev/shm
/dev/dasdb1 6967656 4132924 2474128 63% /root
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.001 MB perf.data (~29 samples) ]
```

■ Display the collected sample data

```
# perf report
```

IBM z Systems kernel features – core improvements

■ **Add interface for partition-resource management (kernel 3.14)**

- The diagnose 0x304 interface is used to inspect and change the different LPAR partition-resource parameters
- The LPAR needs to be authorized to participate in CPU management
- The binary kernel interface allows a system management software to control the partition weight and partition-capping flags

■ **Increase number of CPUs and limit memory use (kernel 3.15)**

- zEC12 can have up to 101 CPUs in an LPAR, z13 up to $141 * 2 = 282$ (with SMT)

■ **Spinlock refactoring (kernel 3.16)**

- Do a load-and-test first, before trying to get the lock with compare-and-swap, this is required for the SMT support with z13
- Improve code generation of the inline spinlock code

■ **Add diagnose 0x288 watchdog support for LPAR (kernel 3.16)**

- The diagnose 0x288 watchdog driver has been converted to the generic watchdog API
- The LPAR variant of the diagnose 0x288 watchdog has been added

IBM z Systems kernel features – core improvements

- **Feed randomness with system information (kernel 3.17)**
 - Use the virtual-machine CPU information data block and the CPU-ID of the boot CPU as a source of device randomness.
 - Helps with KVM to provide a different initial random pool for each virtual machine
- **Architecture backend for uprobes (kernel 3.18)**
 - Uprobes implements a mechanism by which a kernel function can be invoked whenever a process executes a specific instruction location
 - E.g. trace all callers of the user space libc function 'malloc'
- **Improved ftrace/kprobes architecture backend (kernel 3.19)**
 - Switching on CONFIG_FTRACE has a performance impact
 - Rework the function prolog for ftrace/kprobes to reduce the cost
 - With the new code ftrace/kprobes can be used in a production kernel
- **HMC drive CD/DVD access (kernel 3.18)**
 - The HMC device driver allows user space access to a HMC media drive
 - Can be used in a z/VM and an LPAR environment

IBM z Systems kernel features – I/O improvements

- **Lazy IOTLB flushing for PCI DMA unmap (kernel 3.17)**
 - Delay the costly IOTLB flush operation by keeping track of DMA addresses after unmap
 - Only after the DMA addresses have wrapped the IOTLB is flushed
- **DASD support for control unit initiated reconfiguration (kernel 3.18)**
 - A storage server can indicate concurrent hardware changes to the OS
 - Automates channel path management for selected DS8000 service actions
- **Auto port scan resiliency for zfcpl (kernel 3.19)**
 - Improves the Fibre Channel port scan behaviour
 - Avoids concurrent port scans in a virtual server environment with a short delay
 - Rate limit the number of ports scans
- **Handle multiple SCM requests in one HW requests (kernel 3.19)**
 - Do up to 8 block layer requests per HW request to improve performance
- **PCI memory access system calls (kernel 3.19)**
 - Emulate PCI memory access from user space by using system calls

IBM z Systems kernel features – networking & security

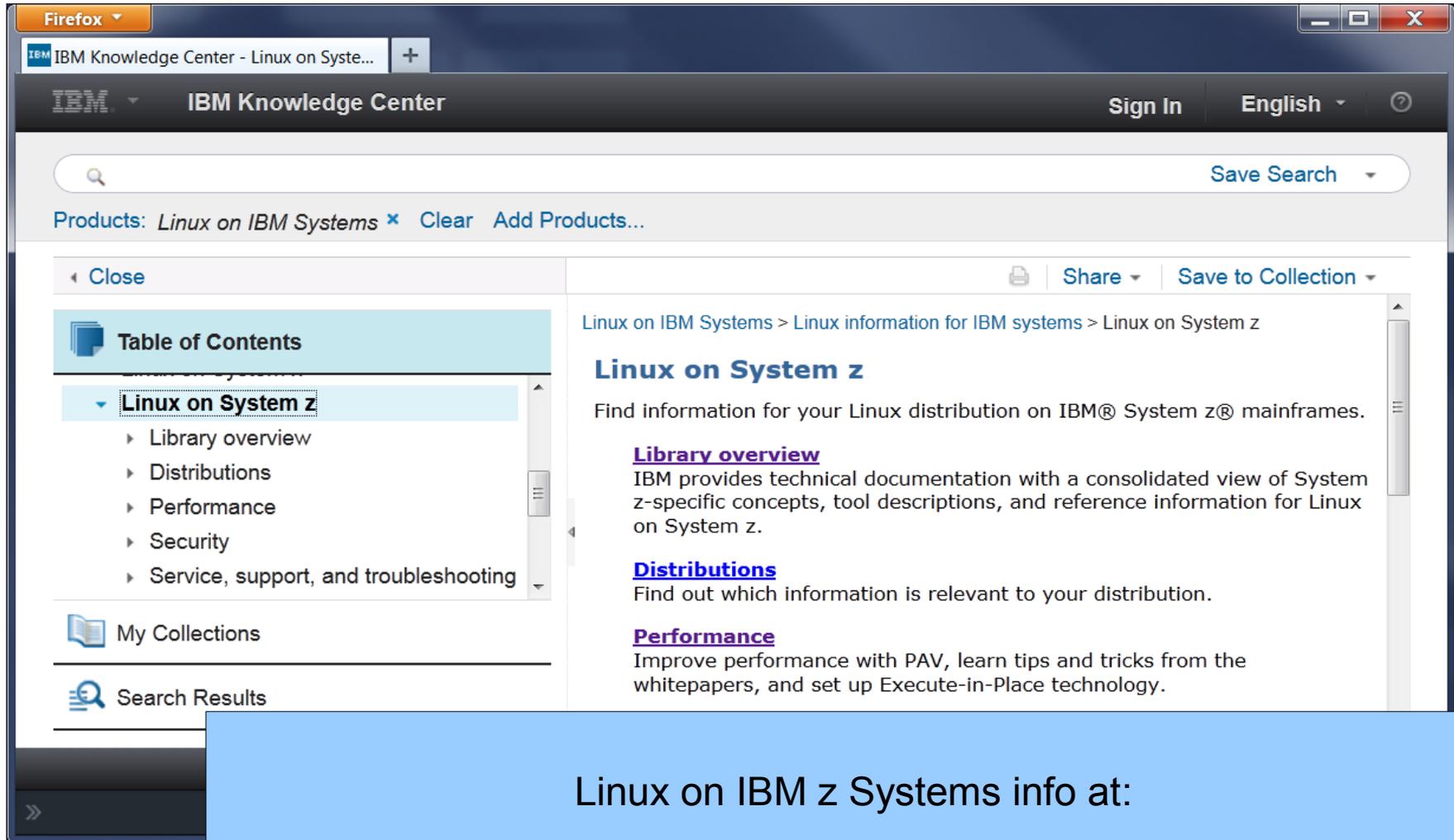
- **HiperSockets layer 2 bridge port functionality (kernel 3.14)**
 - With Linux acting as a software network bridge the network port acting as the bridge needs to be able to receive frames addressed to unknown MAC addresses
 - HiperSocket devices can be configured as primary and secondary bridge ports
- **qeth: priority queueing options + IPv6 support (kernel 3.16)**
 - Adds new options for priority queueing
 - SKB priority queue
 - VLAN priority queueing based on the data in the 802.1Q header
- **Add support for EP11 coprocessor cards (kernel 3.14)**
 - Extend the zcrypt driver with a new capability to service EP11 requests for the Crypto Express4S card in EP11 (Enterprise PKCS#11 mode) coprocessor mode
 - For more information about EP11, see “*Exploiting Enterprise PKCS #11 using OpenCryptoki*”, SC34-2713
- **Support for extended number of AP domains (kernel 3.18)**
 - Extends the number of AP domains recognized by the zcrypt device driver to 256

IBM z Systems kernel removals

- **31-bit kernel support will be removed with kernel 4.1**
 - 31-bit application will continue to work, only the 31-bit kernel is discontinued
 - A 64-bit kernel can be used in combination with a 100% 31-bit user space

- **The claw network driver will be removed with kernel 4.1**
 - Common Link Access for Workstation, used some old RS/6000s, Cisco Routers (CIP) and 3172 devices
 - We could not find a single user of this device driver

New - ibm.com/support/knowledgecenter/



Firefox

IBM Knowledge Center - Linux on System z

IBM Knowledge Center Sign In English

Search Save Search

Products: *Linux on IBM Systems* × Clear Add Products...

Close

Table of Contents

- Linux on System z
 - Library overview
 - Distributions
 - Performance
 - Security
 - Service, support, and troubleshooting

My Collections

Search Results

Linux on IBM Systems > Linux information for IBM systems > Linux on System z

Linux on System z

Find information for your Linux distribution on IBM® System z® mainframes.

Library overview
IBM provides technical documentation with a consolidated view of System z-specific concepts, tool descriptions, and reference information for Linux on System z.

Distributions
Find out which information is relevant to your distribution.

Performance
Improve performance with PAV, learn tips and tricks from the whitepapers, and set up Execute-in-Place technology.

Linux on IBM z Systems info at:

ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_main.html

Documentation news – Updates available

■ Linux Distributions

- SUSE Linux Enterprise Server 11 SP 3
- Red Hat Enterprise Linux 6.4

■ Upstream Linux 3.14, 3.16, 3.17

- ibm.com/developerworks/linux/linux390/documentation_dev.html

■ Security

- Secure Key Common Cryptographic Architecture
4.2.10 Application Programmer's Guide
- Libica 2.2.0. Programmer's Guide

■ Redbooks

- IBM Wave, Virtualization, Oracle, Security

■ Whitepapers

- FileNet P8 5.1, Live Guest Relocation, iSCSI

Linux on System z



Device Drivers, Features, and Commands

Development stream (Kernel 37)

Linux on System z



Using the Dump Tools



Security for Linux on System z

Linux on System z



How to Improve Performance with PAV

Development stream (Kernel 26.35)



Linux on System z



Kernel Messages

Development stream (Kernel 37)

s390-tools package: what is it?

- s390-tools is a package with a set of user space utilities to be used with the Linux on IBM z Systems distributions.
 - It is **the** essential tool chain for Linux on IBM z Systems
 - It contains everything from the boot loader to dump related tools for a system crash analysis .

- This software package is contained in all major (and IBM supported) enterprise Linux distributions which support s390
 - RedHat Enterprise Linux version 4, 5, 6, and 7
 - SuSE Linux Enterprise Server version 9, 10, and 11

- Website:
<http://www.ibm.com/developerworks/linux/linux390/s390-tools.html>

- Feedback: linux390@de.ibm.com

s390-tools package: the content

chccwdev chchp chreipl chshut chcrypt chmem CHANGE	dasdfmt dasdinfo dasdstat dasdview fdasd tunedasd DASD	dbginfo dumpconf zfcpdump zfcpdbf zgetdump scsi_logging_level DUMP & DEBUG
lscss lschp lsdasd lshmc lsluns lsqeth lsreipl lsshut lstape lszcrypt lszfcp lsmem DISPLAY	mon_fsstatd mon_procd ziomon hyptop MONITOR	vmconvert vmcp vmur cms-fuse z/VM
hmcdrvfs zdsfs FILESYSTEM	ip_watcher osasnmpd qetharp qethconf qethqoat NETWORK	cpuplugd iucvconn iucvty ts-shell ttyrun MISC
	tape390_display tape390_crypt TAPE	zipl BOOT

s390-tools package: the zdsfs file system

▪ **zdsfs – mount a z/OS DASD**

- The zdsfs file systems is based on FUSE
- Use the zdsfs command to mount a z/OS DASD as read-only Linux file system.
- The zdsfs file system translates the z/OS data sets into Linux semantics
- Allows Linux to read z/OS physical sequential data sets and partitioned data sets
 - Physical sequential data sets are represented as files
 - Partitioned data sets are represented as directories that contain the PDS members as files
- Other z/OS data set formats are not supported
- To access a z/OS DASD the raw-track access mode of the DASD must be enabled

s390-tools package: the hmcdrv file system

▪ **hmcdrvfs – mount the HMC media drive as virtual filesystem**

- The hmcdrvfs file systems is based on FUSE
- On the HMC, the media must be assigned to the associated system image
- Use the hmcdrvfs command to mount the HMC media drive as read-only Linux file system
- Allows Linux to read files from the HMC media drive
- Required privileges
 - For z/VM the guest virtual machine must have at least privilege class B and the media must be assigned to the LPAR where z/VM runs
 - For LPAR, the activation profile must allow issuing SCLP requests.
- One use case would be installation of a distribution from the HMC DVD drive

▪ **lshmc – list media contents in the HMC media drive**

- Useful to quickly check what kind of HMC media drive is attached

Kernel news – Common code

▪ **Linux version 3.14 (2014-03-30)**

- Deadline scheduling class for better real-time scheduling
- zram memory compression mechanism considered stable
- Btrfs inode properties
- Userspace locking validator
- TCP automatic corking

▪ **Linux version 3.15 (2014-05-08)**

- Improved working set size detection
- New file locking scheme: open file description locks
- Faster erasing and zeroing of parts of a file
- File cross-renaming support
- FUSE improved write performance

Kernel news – Common code

▪ **Linux version 3.16 (2014-08-03)**

- Unified Control Group hierarchy
- XFS free inode btree, for faster inode allocation
- TCP Fast Open server mode on IPv6 support
- seccomp: use internal BPF JIT to speed up filters
- btrfs improvements

▪ **Linux version 3.17 (2014-10-05)**

- File sealing with `fcntl()` to eases handling of shared memory
- Secure generation of random numbers with the `getrandom()` syscall
- Support for page fault tracing in perf trace
- Allow seccomp to set a filter across all threads

Kernel news – Common code

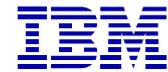
■ **Linux version 3.18 (2014-12-07)**

- Overlayfs to combine two filesystems into a single file system namespace
- bpf() syscall for eBPF virtual machine programs
- TCP: Data Center TCP congestion algorithm
- Networking performance optimization: transmission queue batching

■ **Linux version 3.19 (2015-02-08)**

- Btrfs: support scrubbing and fast device replacement in RAID 5 and 6
- New architecture: Altera Nios II processors
- Networking: support for routing and switching offloading
- NFSv4.2 support for hole punching and preallocation

Questions?



Martin Schwidefsky

*Linux on IBM z Systems
Development*

*Schönaicher Strasse 220
71032 Böblingen, Germany*

*Phone +49 (0)7031-16-2247
schwidefsky@de.ibm.com*