News on the Linux kernel side: Important changes for zSeries

Christian Bornträger
IBM LAB Böblingen Germany
Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM *  
z/VM *
VM/ESA *  
zSeries *
S/390 *

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.
Intel is a trademark of the Intel Corporation in the United States and other countries.
Linux is a registered trademark of Linus Torvalds
Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.
Penguin (Tux) compliments of Larry Ewing.
UNIX is a registered trademark of The Open Group in the United States and other countries.

* All other products may be trademarks or registered trademarks of their respective companies.
Agenda

- Development Process of Linux on zSeries
- Linux 2.6 overview
- Detailed look at zSeries
- Summary
- Discussion
The Linux on zSeries Development Model

- **Kernel 2.4 model**
  - Code drop on developerWorks
  - Vanilla kernel (kernel.org) has been often not up-to-date

- **New model**
  - Code drop is still available and serviced
  - For production use, we suggest to use a distribution kernel
  - As of 2.5 kernels the vanilla kernel has been much more up-to-date and contains most of the changes from the latest code drop
  - It is possible to use the latest vanilla kernel for custom test kernels or as a base for distributions
What is new in Linux 2.6

- There are lots of new features in 2.6....
  - O(1) scheduler, kernel preemption
  - New device model / sysfs
  - Improved scalability / locking
  - More users, groups, PIDs
  - Networking: epoll, IPSEC
  - Threading: NPTL
  - More file systems, Access Control Lists
  - Asynchronous I/O
  - .....
What is new after Linux 2.6.0?

...And several new items after 2.6....

- CPU hotplug
- Block Device Layer
  - I/O barrier support
  - Scalability: per backing dev-unplugging
  - CFQ disk I/O scheduler
- Snapshot, and mirroring in the device mapper
  - session about device mapper->check the updates
- 4k kernel stacks
- Object-based reverse mapping VM
Impact?

- **...So, what items are important for zSeries?**
  - Not all features affect zSeries
    - IDE layer update
    - Desktop interactivity work
  - Some changes help but are not that important
    - O(1) scheduler
    - Kernel preemption (deactivated on most distributions)
  - Other changes affect zSeries
    - Device model
    - Memory management
    - Block device layer
    - Code cleanups
Device Model

- **Device handling in Linux 2.4**
  - Linux 2.4 and earlier kernels have no deep knowledge about the hardware, the attached devices and their relationship
  - Device drivers and subsystems are responsible for handling systems
  - Issues with power management on x86 (suspend and resume) as well as hotplug

- **Introduction of a completely new device model**
  - Linux 2.6 has a hierarchical view about all devices
  - The kernel offers several views on the hardware
Sysfs

- The internal representation is exported via sysfs
- Sysfs is a virtual file system, often mounted on /sys
- All attached devices are visible in sysfs

```
sys/
|-- block
|-- bus
|-- class
|-- devices
|-- firmware
|-- module

/sys/devices/css0
|-- 0.0.0000
|   |-- 0.0.5c18
|      |-- 0.0.0001
|      |-- 0.0.5c19
|      |-- 0.0.0002
|      |   |-- 0.0.f572
|      |   |-- 0.0.0003
|      |   |-- 0.0.f573
|      |   |-- 0.0.0004
|      |   |   |-- 0.0.f574
|      |   |   |   |-- 0.0.001c
|      |   |   |      |-- 0.0.5468
|      |   |      |   |-- chp0.40
|      |      |   |   |   |-- chp0.fc
|      |      |   |   | |   |-- chp0.fd
```

/sys/devices/css0/0.0.0000/0.0.5c18/
|-- availability
|-- block -> ../../../block/dasda
|-- cmb_enable
|-- cutype
|-- detach_state
|-- devtype
|-- discipline
|-- online
|-- readonly
|-- use_diag
Linux Common I/O Layer

- All channel devices are driven by channel programs consisting of channel command words
- These devices are called ccw devices in Linux
  - DASD, OSA devices, zFCP devices...: /sys/bus/ccw/
- Some devices have more than one device number: grouped ccw devices (ccwgroup)
  - OSA network adapter.... /sys/bus/ccwgroup/
- Every Device is part of the hierarchy
- Device model offers different views
- Concept of devices, busses and classes
Sysfs

```
$sysfs/
  |-- /sys
  |   |-- class
  |   |   |-- devices
  |   |   |   |-- ccw
  |   |   |   |   |-- ccwgroup
  |   |   |   |   |   |-- subchannel
  |   |   |   |   |   |   |-- devno
  |   |   |   |   |   |   |   |-- drivername
  |   |   |   |   |   |   |   |   |-- devno
  |   |   |   |   |   |   |   |   |   |-- ...
  |   |   |   |   |   |   |   |   |   |   |-- ...
```
Sysfs

- Use sysfs to configure almost every device
- Use the online attribute to enable/disable a device
  - echo 1 > /sys/bus/ccw/devices/0.0.0190/online
  - echo 0 > /sys/bus/ccw/devices/0.0.0190/online
- chandev.conf is not supported (and not necessary)
  - There is a conversion tool for SLES9 configuration files
    - /etc/sysconfig/hardware/scripts/chandev-to-hwcfg.sh
  - sysfs provides all means to configure devices

  # echo 0.0.0100,0.0.0101,0.0.0102 > /sys/bus/ccwgroup/drivers/qeth/group
  # echo hw_checksumming > /sys/bus/ccwgroup/devices/0.0.0100/checksumming
  # echo 1 > /sys/bus/ccwgroup/devices/0.0.0100/online
Sysfs

- To ease the use, IBM provides helper scripts
  - **lsdasd**
    - Similar to “cat /proc/dasd/devices”
  - **lscss**
    - Similar to “cat /proc/subchannels”
  - **lstape**
    - To show tape devices
  - **chccwdev**
    - For enabling (-e) and disabling devices (-d)
Sysfs - distributions

- configuration via sysfs is usually made by your distribution

- SUSE SLES9
  - hwup and hwdown using the config files in “/etc/sysconfig/hardware”
  - ifup and ifdown using the config files in “/etc/sysconfig/network”
Hotplug

- **Automation the reaction on hardware changes**
  - In case of hardware events, the kernel calls a program or script \"/sbin/hotplug\" with several parameters.
  - Events are for example device add and removal.
  - \(/sbin/hotplug\) is a multiplexer, which calls several agents.
  - The administrator can configure hotplug to automate Linux.

- **Hotplug is integral part of the kernel**
  - In Linux 2.4, every device driver has been responsible for the creation of hotplug events: lots of duplicated code.
  - In Linux 2.6, every device drivers which uses the device model gets hotplug for free.
udev

- **User can access devices via device nodes**
  - To read the first DASD partition you can read `/dev/dasda1`
  - Device nodes are normal files, which need to be created
    - Static device nodes
      - Created by the administrator
    - Devfs (deprecated)
      - Kernel file system
      - Device nodes are created by the kernel
      - How to define the policy?
    - Udev:
      - Application, that creates device nodes using hotplug and sysfs
      - Udev is the proposed way to handle device nodes
      - Possible using the device model
udev – how it works

- The kernel calls `/sbin/hotplug` with parameters
- `/sbin/hotplug` multiplexes events and calls udev
- Environment variables (examples)
  
  ```
  DEVPATH=/class/net/eth1
  PATH=/sbin:/bin:/usr/sbin:/usr/bin
  ACTION=add
  PWD=/
  SHLVL=1
  HOME=/
  INTERFACE=eth1
  SEQNUM=196
  
  DEVPATH=/block/dasdb
  PATH=/sbin:/bin:/usr/sbin:/usr/bin
  ACTION=add
  PWD=/
  SHLVL=1
  HOME=/
  SEQNUM=201
  ```

- `/sys/block/dasdb/dev` contains major and minor number
- `/etc/udev/udev.rules` and `/etc/udev/udev.permissions` define udevs policy which creates a device node using the major number, the minor number

See my session on Tuesday
Memory Management

- How does Linux handle a memory request if memory is exhausted?

Apps

- page fault
- find unused/freeable page
- find all users
- free page frame
- fulfill requirement

physical memory

Memory Management

- How does Linux handle a memory request if memory is exhausted?

Apps

- page fault
- find unused/freeable page
- find all users
- free page frame
- fulfill requirement

physical memory
Memory Management

- **ISSUE:**
  - Finding all users is expensive
  - Kernel has to lookup every process

- **SOLUTION:**
  - Add additional information about every physical page. Feature is called reverse mapping and is available for all platforms
  - There has been a small overhead, which was addressed using objective rmap
Memory Management

- **virtual memory**
  - More optimizations based on rmap
  - Every operating system that provides virtual memory needs to keep track whether memory pages are used to read or write
  - Pages which are written to are called “dirty”
  - The dirty information is usually stored in a bit
    - Per virtual page on x86
    - Per hardware page on zSeries
  - zSeries difference allows optimizations
  - Kernel 2.6 offers the infrastructure to exploit the hardware feature
dbench scalability

SLES 8

SLES 9

Transferrate vs. # of processes for SLES 8 and SLES 9 with different numbers of CPUs.

- SLES 8:
  - 1CPU
  - 2 CPUs
  - 4 CPUs
  - 8 CPUs
  - 16 CPUs

- SLES 9:
  - 1CPU
  - 2 CPUs
  - 4 CPUs
  - 8 CPUs
  - 16 CPUs
Block Device Layer

- **Responsible for block devices**
  - Random access devices
  - Addressable in blocks
  - e.g. DASD, floppy disks, xpram, ....
  - Responsible for optimizing the access to block devices
    - Maximize throughput
    - Minimize latency
    - Fairness
    - There is no optimal algorithm
Block Device Layer - I/O scheduler

- **I/O schedulers can improve the performance**
  - Reorder requests
  - Merge requests

- **It is not that easy...**
  - Competing processes
  - Mixed write and read requests
  - Heuristics are used
Block Device Layer - internals

- **Data transfer is organized in request queues**
  - Input and output goes through request queues
  - Request queues can be plugged (stopped) and unplugged (running)

- **Why plugging?**
  - Only stopped request queues can be optimized
  - I/O scheduler optimizes plugged (stopped) queues
  - Afterwards the queues are unplugged to start the optimized I/O operations
Block Device Layer - news

- The block device layer has been completely rewritten during the 2.5 phase
  - Improved internal data structures
    - Higher flexibility and scalability
  - Unplugging per device instead of global unplugging
  - Larger block devices up to 16TB/8EB (32/64bit)
  - Modular I/O schedulers
    - You can choose the optimization strategy
    - Better performance than 2.4
    - You can choose the I/O scheduler using the elevator kernel parameter (elevator=as,deadline,noop,cfq)
Threading

- **NPTL (Native POSIX Thread Library)**
  - A new user space library boosts performance of multithreaded applications. Several kernel features are used:
    - Thread Local Storage (TLS)
    - Futex ('Fast Userspace muTexes')
  - Transparent for the application (binary compatible)
    - Most applications will run with the new library
    - Nevertheless, there is a change in behavior, to comply with POSIX standard
    - In case of trouble with old applications, try to run with
      ```
      LD_ASSUME_KERNEL=2.4.18 <command>
      ```
CPU hotplug

- It is now possible to set CPUs online / offline
- Access via `/sys/devices/system/cpu/cpu<num>`
- Activate CPUs on the fly
  - `#CP DEFINE CPU 1`
  - `echo 1 > /sys/devices/system/cpu/cpu1/online`
- Deactivate CPUs
  - `echo 0 > /sys/devices/system/cpu/cpu3/online`
  - sends a SIGP STOP to the CPU
- available on SLES9 SP1
System Call Emulation

- It is possible to run 31bit applications on a 64bit system mixed with 64bit applications
- Same functionality for sparc, ppc, x86-64, mips
- Translation of parameters
- Translation of results
- Transparent
System Call Emulation

- **What has been wrong with 2.4?**
  - In Linux 2.4 every architecture has provided its own layer
  - All implementations are inspired by sparc64 code
  - Lots of code duplicates (including errors)
  - Bug fixes have been often applied to one architecture only

- **What’s new in 2.6?**
  - Common code for all architectures has been created
  - Ongoing process of moving the feature into common code
  - Several errors fixed during the consolidation: higher quality
  - Aim: emulated 31 bit exactly as good as native 31 bit
System Call Emulation

- **What do I need to run a 31 bit application on 64bit?**
  - All necessary libraries must be available in 31 bit as well
  - Some mixed JAVA/native code applications need 31 bit JAVA libraries as well
  - Usually 64 bit libraries are `/lib64/`, `/usr/lib64/` ...
  - 31 bit libraries are `/lib/`, `/usr/lib/` ...
  - Use `ldd` to see the dependencies

```bash
# ldd /bin/bash
libreadline.so.4 => /lib64/libreadline.so.4 (0x0000010000021000)
libhistory.so.4 => /lib64/libhistory.so.4 (0x00001000063000)
libncurses.so.5 => /lib64/libncurses.so.5 (0x0000100006c000)
libdl.so.2 => /lib64/libdl.so.2 (0x000010000d4000)
libc.so.6 => /lib64/libc.so.6 (0x000010000d8000)
/lib/ld64.so.1 => /lib/ld64.so.1 (0x00001000000000)
```
On Demand Timer Patch

- **Linux uses a regular timer for internal work**
  - 100 or 1000 ticks per second
  - Timer tick has a relevant overhead having many guests

- **IBM provided a patch to deactivate the timer on idle systems**
  - Integrated into SLES8 and SLES9
  - Since 2.6.6 part of the standard Linux kernel
  - `/proc/sys/kernel/hz_timer`
    - Set to 0 to deactivate the regular tick (patch enabled)
    - Set to 1 to activate the regular tick (default, patch disabled !)
zipl – some news besides the kernel

- Zipl allows to define a boot menu
  - You can define several configurations containing of all possible options
    - Kernel
    - Initrd
    - Parameters

- Configuration

```
[defaultboot]
defaultmenu = menu

[ipl]
target = /boot/zipl
image = /boot/image
ramdisk = /boot/initrd
parameters = "root=/dev/dasda1 selinux=0 TERM=dumb elevator=cfq"

[test]
target = /boot/zipl
image = /boot/image-test
ramdisk = /boot/initrd-test
parameters = "root=/dev/dasda1 dasd=1234 selinux=0 TERM=dumb"
```

:menu
1=ipl
2=test
target=/boot/zipl
default=1
timeout=10
prompt=1
You can choose the configuration at boot

```plaintext
00: zIPL v1.3.1 interactive boot menu
00:
00:   0. default (ipl)
00:
00:   1. ipl
00:
00:   2. test
00:
00: Note: VM users please use 'cp vi vmsg <input>'
00:
00: Please choose (default will boot in 10 seconds):
```

You can also pass kernel parameters

- e.g.: `cp vi vmsg 1 dasd=1000-1fff`

Ideal for recovery and testing
Distributions

- Which distributions offer Kernel 2.6 support?
  - SUSE SLES9 : GA August 2004
  - Red Hat RHEL4: GA 2005
  - Debian unstable

- Other distributions will follow

- SUSE Linux supports upgrade SLES8 to SLES9
Outlook

- Linux 2.6 will see lots of improvements without any Linux 2.7
  - Currently 10 MB/month of patches
- Open source, IBM and other companies are developing more enterprise features
  - CKRM: class based resource manager, something like z/OS WLM
- Better integration with z/VM
- This presentation will probably look quite different in a year
There are lots of features and changes

This presentation give some details on some aspects

Please ask me if:
  – you want to know more about a specific feature
  – you miss some feature
  – you think we do something completely wrong
Summary

- Cleanups
- Rewrites
- Limits have been lifted
- Scalability was increased
- New device model
- New features
Question & Discussion

- Now

- I am available afterwards
  - After this session
  - Any time during WAVV
  - Email: cborntra@de.ibm.com

- Thank you for your attention

Backup Slides
Virtual Memory

- **Dirty bit handling**
  - Pages can be dirty or clean
  - If you remove a dirty page, you have to write the page back
  - The dirty/clean bit is usually stored in page table entries (per address space). To get the dirty information for a physical page, Linux used to query all processes
  - S/390 zSeries stores the dirty information in storage keys
  - Storage keys are already per physical page!
  - Optimization which led to drastic reduction in SSKE use
Kernel Build

- For production use, distribution kernels suggested
  - Support is available
  - 3590 OCOs are available
  - Well tested and serviced

- For testing new features a kernel.org kernel will probably work fine
  - No support
  - No 3590 OCOs
Architecture merge s390/s390x

- **Linux has 2 possible modes:**
  - 31bit, called s390 and 64bit, called s390x

- **Both modes were implemented by different Linux architecture code**
  - Lots of duplicated code
  - Possible inconsistencies

- **To improve stability both modes were collapsed into the same C-code**
  - Highly reduced code size!
The Linux Development Model

- **Previous model**
  - Even minor numbers (2.0, 2.2, 2.4..) indicate a stable kernel
  - Odd minor numbers (2.1, 2.3...) indicate an unstable kernel
  - Major developments only in unstable kernels

- **New model**
  - There is a test environment called -mm kernels
  - mm kernel follows the vanilla kernel
  - Proved changes will be merged into the vanilla 2.6 kernel
  - No need for a kernel 2.7 at the moment