IBM IT Education Services

L22
Neale Ferguson

Linux 2.6 – An early peek

zSeries Expo

November 10 - 14, 2003 | Hilton, Las Vegas, NV

Acknowledgements

- Material drawn from:
  - Wonderful World of Linux 2.6 by Joseph Pranevich

    http://kniggit.net/wwol26.html
  - Towards Linux 2.6 by Anand K Santhanam

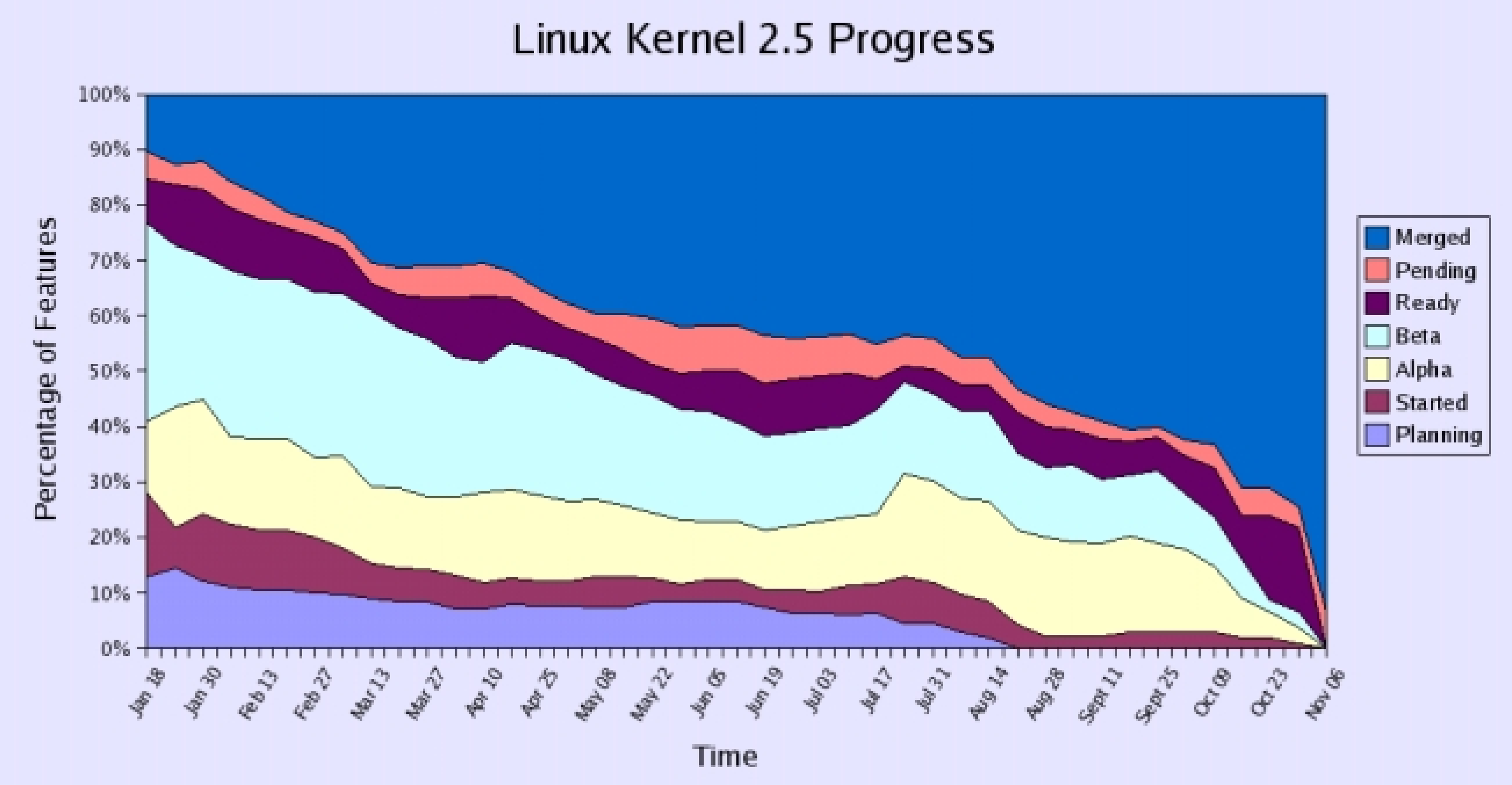
    http://www-106.ibm.com/developerworks/linux/library/l-inside.html
  - What's new in Linux 2.6? By Dr. Ulrich Weigand

    Session L05 at this conference
  - The Native POSIX Thread Library for Linux by Ulrich Drepper and Ingo Molnar

Agenda

- A brief overview of 2.6 features
  - Kernel Features
  - Platforms
  - Scheduler & Preemption
  - Network, Filesystem, and Scalability
- A closer look at:
  - NPTL
  - New device filesystem – sysfs
  - Device Drivers
  - Kernel Building

Kernel Features



Platform and Device Support

- New architectures
  - PowerPC 64-bit (ppc64)
  - AMD 64-bit (x86_64)
  - µcLinux (MMU-less processors: v850, m68knommu)
  - User Mode Linux
- New devices
  - New input device / frame buffer layers
  - ALSA (Advanced Linux Sound Architecture)
  - Video for Linux v2
  - New IDE layer, Serial ATA support

New Scheduler

- In 2.4:
  - Timeslice recalculation algorithm requires that all processes exhaust their timeslice before their new timeslices can be recomputed
  - Affects performance of SMP systems as processes idle while waiting for recalculation of timeslice
  - Processes can bounce between CPUs
- In 2.6:
  - Timeslices are distributed on a per-CPU basis: eliminating global synchronization and recalculation
  - Scheduler maintains a per-processor run queue/lock mechanism so that two processes on two different processors can sleep, wake up, and context-switch completely in parallel

New Scheduler – Advertised Benefits

- SMP efficiency: If there is work to be done, all the processors should work.
- Waiting processes: No process should stay without processor time for long periods of time; additionally, no process should take an unreasonably high amount of CPU time
- SMP affinity: Processors should affine to one CPU and will not bounce between CPUs
- Priorities: Less important tasks should start with lower priority (the converse is also true)
- Load balancing: The scheduler will decrease the priority of any process that generates more load than the processor can handle
- Interactive performance: With the new scheduler, the user should not see the system taking longer to respond to things like mouse clicks or key taps, even under very high loads

Kernel Preemption

- A kernel task can be preempted so that some important user process can continue to run
- Critical sections of the kernel are locked against preemption
- Code not complete for zSeries as of 2.6.0-test7

File System Enhancements

- Support for new file systems
  - IBM JFS
  - SGI XFS
  - NFS v4 – not a full implementation
  - Andrew File System (AFS) – read only mode
  - NTFS r/w – “less experimental”
- Other enhancements
  - Device mapper infrastructure (LVM2, EVMS)
  - Extended Attribute / Access Control List (ACL) support
  - Large directory support for ext2/ext3
  - Zero-copy NFS

Networking Enhancements

- **`/dev/epoll`** enables applications specifically tailored to detect and use it to operate more quickly
- A new device in the kernel, **`/dev/epoll`**, allows programmers to efficiently enumerate pending events on a number of sockets or pipes
- It works in a manner somewhat similar to **`poll()`** and is used in a very similar fashion to Solaris 8's **`/dev/poll`** device
- Exploited by Notes to handle thousands of connections with minimal overhead

Networking Enhancements

- IPSec
  - Collection of protocols for IPv4 and IPv6
  - Security at the protocol layer – no need for application to be aware of it
  - Similar in concept to SSL but at a much lower level
  - In-kernel encryption support for SHA, DES, and others
- Improved multicast support
  - New SSM protocols: MLDv2, IGMPv3
- vLAN configuration no longer “experimental”

Networking Enhancements

- NFSv4
  - Subset of functions
  - Stronger and more secure authentication with cryptography (a kernel based crypto API is available)
- NFS
  - Up to 64 times as many concurrent users and larger request queues
  - **`lockd`** and **`nfsd`** separated
  - Improved support of NFS-shared volumes as the root filesystem

Scalability

- Reduced used of Big Kernel Lock & elimination of global locks
- Per-CPU data structures
- Increased number of threads to 2GB
- Block device limits now 16TB (32 bit) or 8EB (64 bit)
- Major/minor device numbers increased to 4K/1M

NPTL – New POSIX Threads Mechanism

- Replacement for linuxthreads
  - Manager thread required for userland implementation causes creation and cleanup problems
  - Signal system is not POSIX compliant and leads to several problems
  - Each thread has a different process ID
  - Large multi-threaded applications (e.g. Java based) may create thousands of threads - /proc system becomes almost unusable

NPTL – New POSIX Threads Mechanism

- Goals of new mechanism
  - POSIX compliance
  - Effective use of SMP
  - Low overhead for creation and cleanup
  - Binary compatibility with existing applications
  - Scalability
  - Integration with C++

NPTL – New POSIX Threads Mechanism

- Design points
  - 1:1 rather than m:n
  - Kernel to implement POSIX signal handling
  - Elimination of the manager thread
  - Kernel implementation of synchronization primitives
    - Introduction of the futex (fast mutex)
  - Optimized memory allocation

NPTL – New POSIX Threads Mechanism

- Kernel enhancements
  - Support of an arbitrary number of thread-specific data area
  - The **`clone()`** system call extended to optimize thread creation
  - POSIX signal handling:
    - Signals sent to the process are now delivered to one of the available threads
    - Fatal signals terminate the entire process
    - Stop and continue signals affect the entire process
    - Shared pending signals are supported

NPTL – New POSIX Threads Mechanism

- Kernel enhancements
  - An **`exit_group()`** system call introduced to terminate an entire process, **`exit()`** terminates the current thread (this call has been optimized)
  - **`exec()`** now provides the newly created process with the ID of the original
  - Entire process resource usage reported to the parent
  - Support for detached threads
  - Kernel keeps the initial thread around until all threads have exited

NPTL – Obvious Differences - Old

```
> ps –u usanefe -wf
usanefe   4234  4210  0 14:57 pts/0    00:00:00 -csh
usanefe   6704  4234  0 16:41 pts/0    00:00:00 ./ThrCancel
usanefe   6705  6704  0 16:41 pts/0    00:00:00 ./ThrCancel
usanefe   6706  6705  0 16:41 pts/0    00:00:00 ./ThrCancel
usanefe   6707  6705  0 16:41 pts/0    00:00:00 ./ThrCancel
usanefe   6708  6705  0 16:41 pts/0    00:00:00 ./ThrCancel
usanefe   6710  6705  0 16:41 pts/0    00:00:00 ./ThrCancel

> ls /proc/6705
auxv  cmdline  cwd  environ  exe  fd  maps  mem  mounts  root  stat
statm  status  task

> ls /proc/6705/task
6705
```

NPTL – Obvious Differences - New

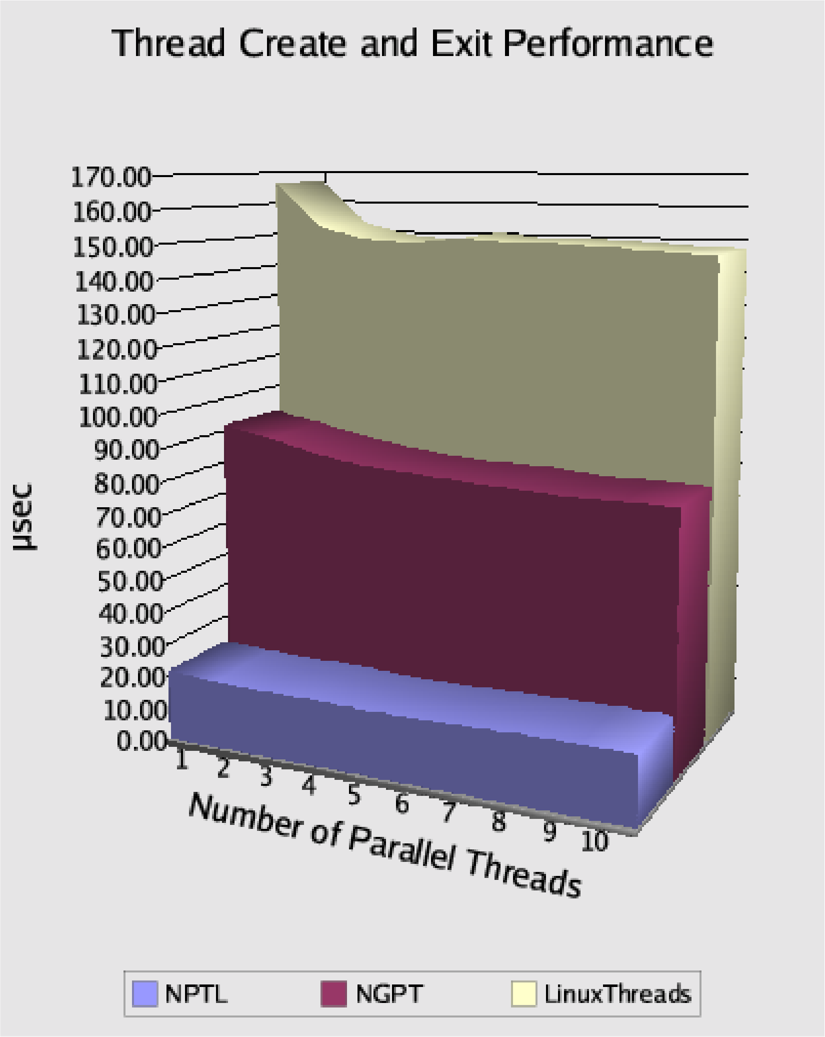
```
> ps -u usanefe -wf
UID        PID  PPID  C STIME TTY          TIME CMD
usanefe    661   638  0 16:45 pts/0    00:00:01 -csh
usanefe    680   661  0 16:47 pts/0    00:00:00 ./ThrCancel

> ls /proc/680
auxv  cmdline  cwd  environ  exe  fd  maps  mem  mounts  root  stat
statm status  task

> ls /proc/680/task
680 681  682  683  687

> ls /proc/680/task/687
auxv  cmdline  cwd  environ  exe  fd  maps  mem  mounts  root  stat
statm status
```

NPTL –Comparison



NPTL – Minor Incompatibility

- According to Single UNIX Specification:
  - **`semop()`** *may* be a cancellation point
  - Under linuxthreads it is
  - Under NPTL it is not
  - Therefore if you issue a **`pthread_cancel()`** using the “deferred” option a thread waiting on the **`semop()`** operation will not be woken and cancelled

FUTEXes

- Futexes are a way multiple processes or threads can serialize events so they avoid "race conditions”
- Unlike the traditional mutex operations this is partially kernel based (but only in the contention case)
- It supports setting priorities to allow applications or threads of higher priority access to the contested resource first
- By allowing a program to prioritize waiting tasks, applications can be made to be more responsive in timing-critical areas.

System File System - sysfs

- Model unifies all the current driver models in the kernel
- A visible representation of the device tree as the kernel sees it
- Augments the bus-specific drivers for bridges and devices by consolidating a set of data and operations into globally accessible data structures
- The common device and bridge interface facilitates seamless plug-and-play, power management, and hot plug
- Exports the hierarchical view of all devices to userland

Sysfs - Contents

```
/sys/block/sda:
dev  device  queue  range  sda1  sda2  size  stat

/sys/block/sda/queue:
iosched  nr_requests

/sys/block/sda/queue/iosched:
antic_expire  read_batch_expire  read_expire
write_batch_expire  write_expire

/sys/block/sda/sda1:
dev  size  start  stat

/sys/block/sda/sda2:
dev  size  start  stat
```

Sysfs – Device Representation

```
0.0.000f
    Attributes:
        detach_state   : 0
        chpids : 05 11 00 00 00 00 00 00
        pimpampom      : c0 c0 ff
      0.0.034d
          Attributes:
              detach_state     : 0
              devtype  : 3390/0a
              cutype   : 3990/e9
              online   : 1
              readonly : 0
              discipline       : ECKD
              use_diag :
```

Sysfs – Adding Devices

```
insmod /lib/modules/`uname -r`/kernel/drivers/s390/cio/ccwgroup.ko 2>/dev/null
insmod /lib/modules/`uname -r`/kernel/drivers/s390/net/qeth_mod.ko 2>/dev/null
echo "0.0.0900,0.0.0901,0.0.0902" > /sys/bus/ccwgroup/drivers/qeth/group
echo "VOSASW" > /sys/bus/ccwgroup/drivers/qeth/0.0.0900/portname
echo 1 > /sys/bus/ccwgroup/drivers/qeth/0.0.0900/online

insmod /lib/modules/`uname -r`/kernel/drivers/s390/scsi/zfcp.ko 2>/dev/null
echo "0x5005076300cfa20a" >/sys/devices/css0/0.0.0012/0.0.d008/port_add
echo "0x5403000000000000" >/sys/devices/css0/0.0.0012/0.0.d008/0x5005076300cfa20a/unit_add
echo "1" >/sys/devices/css0/0.0.0012/0.0.d008/online
```

Sysfs – Result of Adding Devices

```
0.0.0009
    Attributes:
        detach_state   : 0
        chpids : 17 00 00 00 00 00 00 00
        pimpampom      : 80 80 ff
      0.0.0900
          Attributes:
              detach_state     : 0
              devtype  : 1732/01
              cutype   : 1731/01
              online   : 1
```

Sysfs – Using systool to Examine

```
> systool -a -v -r css0
Root Device Tree: css0
   css0
       Attributes:
           detach_state : 0
         0.0.0012
             Attributes:
                 detach_state   : 0
                 chpids : 00 00 00 00 00 00 00 00
                 pimpampom      : 80 80 ff
               0.0.d008
                   Attributes:
                       detach_state     : 0
                       failed   : 0
                       in_recovery      : 0
                       port_remove      : store method only
                       port_add : store method only
                       wwnn     : 0x5005076400c98574
                       wwpn     : 0x5005076401003c58
                       s_id     : 0x010900
                       hw_version       : 0x0002
                       lic_version      : 0x00000024
                       fc_link_speed    : 2 Gb/s
                       fc_service_class : 3
                       fc_topology      : fabric
                       scsi_host_no     : 0x0
                       status   : 0x5400002e
                       devtype  : 1732/03
                       cutype   : 1731/03
                       online   : 1
                     nameserver
                         Attributes:
                             detach_state       : 0
                             failed     : 0
                             in_recovery        : 0
                             status     : 0x00000019
                             wwnn       : 0x0000000000000000
                             d_id       : 0xfffffc
```

```
host0
    Attributes:
        detach_state       : 0
      0:0:1:1
          Attributes:
              detach_state : 0
              fcp_lun      : 0x5403000000000000
              wwpn : 0x5005076300cfa20a
              hba_id       : 0.0.d008
              device_blocked       : 0
              queue_depth  : 32
              type : 0
              scsi_level   : 4
              vendor       : IBM
              model        : 2105800
              rev  : .459
              online       : 1
              rescan       : store method only
              delete       : store method only
0x5005076300cfa20a
    Attributes:
        detach_state       : 0
        failed     : 0
        in_recovery        : 0
        status     : 0x54000003
        wwnn       : 0x5005076300c0a20a
        d_id       : 0x010800
        unit_add   : store method only
        unit_remove        : store method only
        scsi_id    : 0x1
      0x5403000000000000
          Attributes:
              detach_state : 0
              scsi_lun     : 0x1
              failed       : 0
              in_recovery  : 0
              status       : 0x54000000
```

Device Driver Changes

- ELF capabilities used to initialize modules
  - **`module_init`** and **`module_exit`** exist within special sections of the ELF object
  - Init and clean-up code called directly by kernel
- No need for use of MOD_DEC/INC_USE_COUNT
  - This is taken care of outside the module
  - Code referencing the module uses **`try_module_get(&module)`** to access the module
- Object is a “kernel object” with a suffix of “.ko”

Kernel Building

- Configuration the same apart from new features
- No need to "**`make dep`**"
- **`make`** is not verbose by default
- **`make subdir/`** will compile all the files within subdir/ and below
- **`make help`** will provide the make targets supported

Kernel Building

```
make[1]: `arch/s390/kernel/asm-offsets.s' is up to date.
  CHK     include/linux/compile.h
  CC      arch/s390/kernel/init_task.o
  CPP     arch/s390/kernel/vmlinux.lds.s
  GEN     .version
  CHK     include/linux/compile.h
  UPD     include/linux/compile.h
  CC      init/version.o
  LD      init/built-in.o
  LD      vmlinux
  OBJCOPY arch/s390/boot/image
  Building modules, stage 2.
  MODPOST
```

Current and Future Work

- Bug in PFAULT handling
  - Can get into a state where interrupts are re-enabled and a page that is just about to be marked as unavailable is flagged as available
  - Circumvention is to set the nopfault parameter
- Running compliance tests
- Move one of our products to see how (if) it will run
- Ported cpint to conform to the new device driver standards (and play well with sysfs)
- Play with preemption when zSeries fixes are in