Linux Kernel 2.6: (3.0?) New Features - II

Jerry Cooperstein

Axian Inc

coop@axian.com

@ OGI 10/15/02

Linux Kernel 2.6
(or 3.0?)

- Feature Freeze: Halloween 2002
- Better Performance, Especially on **SMP**
- Better Scalability
- Better I/O Subsystem, New Filesystems
- Many New Hardware Drivers
- New Platforms
- Many Features Tested as 2.4 Patches

Latest Status

- Guillaume Boissiere maintains a status report, updated weekly at:

http://kernelnewbies.org/status/latest.html

New Features: General (Review of Lecture I)

- Preemptable Kernel
- O(1) Scheduler
- New Kernel Device Structure (**`kdev_t`**)
- Improved Posix Threading Support (NGPT and NPTL)
- New Driver Model & Unified Device Structure

New Features: General (Review of Lecture I)

- Faster Internal Clock Frequency
- Paring Down the BKL (Big Kernel Lock)
- Better in Place Kernel Debugging
- Smarter IRQ Balancing
- ACPI Improvements
- Software Suspend to Disk and RAM

New Features: General (Review of Lecture I)

- Support for USB 2.0
- ALSA (Advanced Linux Sound Architecture)
- LSM (Linux Security Module)
- Hardware Sensors Driver (lm-sensors)

New Features: Architectures (Review of Lecture I)

- AMD 64-bit (x86-64)
- PowerPC 64-bit (ppc64)
- User Mode Linux (UML)

New Features: General

- CPU Clock and Voltage Scaling
- Setting Processor Affinity
- Improved **NUMA** Support
- Reverse Mapping VM System (**rmap**)
- Large Page Support
- High Resolution Posix Timers
- New Serial Port Driver Rewrite and API

New Features: Journalling Filesytems

- Ext3 (already in 2.4)
- ReiserFS (already in 2.4)
- JFS (IBM)
- XFS (SGI)

New Features: I/O Layer

- Rewrite of Block I/O Layer (BIO)
- Asynchronous I/O
- IDE Layer Update
- ACL Support (Access Control List)
- New NTFS Driver

Removed Features

- Export of **`sys_call_table`**
- End of **Task Queues**

New Features: Networking (To be in Lecture III)

- NFS v4
- Zero-Copy NFS
- TCP Segmentation Offload
- SCTP Support
  (Stream Control Transmission Protocol)
- Bluetooth Support (not experimental)
- NAPI (Network Interrupt Mitigation)

CPU Clock and Voltage Scaling

- Change CPU clock speed on the fly
- Save battery power
- Many platforms including: Intel SpeedStep, Transmeta Crusoe, Intel Xeon, AMD PowerNow K6, ARM, AMD Elan, VIA Cyrix Longhaul
- Read and change from **`/proc/cpufreq`**
- **http://www.brodo.de/cpufreq**

Setting Processor Affinity

- **Bind** (or **pin**) a process to specific CPU
- Can set by writing a mask to **`/proc/[pid]/affinity`**
- Or use new system calls:

```
sched_setaffinity(pid,len,&mask);
sched_getaffinity(pid,len,&mask);
```

- **http://www.kernel.org/pub/linux/kernel /people/rml/cpu-affinity**

NUMA Improvements (**N**on-**U**niform **M**emory **A**ccess)

- Scales better for many CPU's than **SMP** (**S**ymmetric **M**ulti-**P**rocessing)
- System may have many **nodes** with: CPU(s), RAM, cache, I/O bus, etc.
- **Local** memory on node running process
- **Remote** memory on other nodes
- Reduces memory bus contention

NUMA Improvements

- **SMP** systems with > 8 processors are **NUMA** internally
- **Hyperthreading** also looks like **NUMA**
- **http://lse.sourceforge.net/numa**

NUMA Improvements

- Discontiguous physical memory patches
  - **CONFIG_DISCONTINGMEM** allows huge holes in physical mem address space
- Scheduler modfications
- Reduced lock contention
- CPU affinity selection

NUMA Improvements: Topology Support

get node containing CPU or memory block:

`__cpu_to_node(cpu)`

`__memblk_to_node(memblk)`

get node containing node:

`__parent_node(node)`

get first CPU in node:

`__node_to_first_cpu(node)`

get bitmask of CPU's on node:

`__node_to_cpu_mask(node)`

get first memory block on the node:

`__node_to_memblk(node)`

Reverse Mapping Virtual Memory System (**rmap**)

- One way mapping:
  - given a virtual address, find page table entry (**PTE**) pointing to page of physical RAM; if not present, generate page fault
  - No inverse operation: find **PTE**'s corresponding to a physical page. This makes freeing memory inefficient. All page tables must be scanned to make sure a page is not referenced.

Reverse Mapping Virtual Memory System (**rmap**)

- Reverse mapping:
  - Create a data structure for each physical page that lists **PTE**'s pointing to it, referenced through the **page** structure
  - Freeing pages much faster, more overhead
  - 2.4 kernel version yanked in 2.4.12; built in piece by piece in 2.5
- (Rik van Riel) **http://surreil.com/patches/**

Large Page Support

(not yet accepted)

- On most architectures, Linux uses 4 KB or 8 KB page frame.
- Many CPU's can work with larger pages; *e.g.*, **x86** can use 4 MB
- Requires fewer **PTE**'s and gets better use of **TLB** (which caches virtual to physical address translations)
- Claims of 30% performance boost

High Resolution Posix Timers

(not yet accepted)

- Add new system calls (instead of glibc):
  `clock_gettime(), clock_settime(), clock_getres(), clock_nanosleep(), timer_settime(), timer_gettime()`
  `timer_create(), timer_delete()`, etc
- George Anzinger:
  http://high-res-timers.sourceforge.net/

New Serial Port Driver Rewrite and API

- Complete redesign
- New low level serial drivers
- New data structures:

  `uart_port, uart_info, serial_struct,`

  `uart_ops, uart_state, uart_driver,` etc.
- New Functions:

  `uart_register,uart_unregister, uart_add_one_port, uart_remove_one_port,` etc

New Serial Port Driver Rewrite and API

- New **`uart_ops`** data structure has entry points:

  `set_mctrl(), get_mctrl(), stop_tx(),start_tx(), stop_rx(), tx_empty(), startup(), shutdown(), release_port(), request_port(), config_port(), break_ctl(), change_speed(), ioctl(),` etc.

Journalling Filesystems

- Operations grouped into **transactions**
- **Transactions** completed **atomically**
- **Log file** records each transaction
- On system failure, power outage, etc., only the most recent transactions need checking
- Result: **fsck** runs very fast (seconds)

Journalling Filesystems: Built into the Kernel

- **EXT3** (in 2.4) Extension of **EXT2,** same on-disk layout, easiest migration path
  http://e2fsprogs.sourceforge.net/ext2.html
- **ReiserFS** (in 2.4) **http://www.namesys.com**
- **JFS (**IBM, AIX) **http://oss.software.ibm.com /developerworks/opensource/jfs**
- **XFS (**SGI, IRIX) **http://oss.sgi.com/projects/xfs**

Journalling Filesystems: Enhancements

- Large files allocated using **extents:**
  (file offset, starting block, length)
- Better handling of large directories
- Dynamic inode allocation
- 64-bit
- Limit internal fragmentation from files smaller than a block

Journalling Filesystems: Features

Feature	Ext3	Reiser	JFS	XFS
Largest Block Size (IA32)	4 KB	4 KB	4 KB	4 KB
Largest Filesystem	16384 GB	17592 GB	18000 PB	32 PB
Largest File Size	2048 GB	1 EB	9000 PB	4 PB
Growing Filesystem Size	Patch	Yes	Yes	Yes
Access Control Lists	Patch	No	Yes	WIP
Dynamic disk inode allocation	No	Yes	Yes	Yes
Data Logging	Yes	No	No	No
Log on external device	Yes	Yes	Yes	Yes

data from Steve Best (IBM), Linux Magazine, October 2002

Rewrite of Block I/O Layer (BIO)

- Complete rewrite
- More tunable at low and high levels
- Better performance possible
- Requires new API for block drivers

Rewrite of BIO Layer: Low Level Tuning

- Per-queue parameters instead of global
  - max request size, sector size, max sectors, etc., can take more optimal values
- High memory I/O support
  - If possible avoid **bounce buffers**
- I/O scheduler modularization
  - Can write a method for a queue, or choose from a list of generic ones

Rewrite of BIO Layer: High Level Tuning

- I/O Barriers
  - Can request strict ordering of requests
  - uses **`BIO_BARRIER`** flag
- Request priority, latency
  - Specify low, med, high priority for request
  - Place latency limits on requests

Rewrite of BIO Layer:

- **Bypass Mode** permits direct low level access without use of **`ioctl's`**
- Larger I/O requests can be sent without fragmenting and then recombining
- **`io_request_lock`** replaced by finer-grained per-queue lock
- 64-bit sector numbers

Asynchronous I/O (AIO)

- Queue up I/O requests, program continues to execute in parallel
- Completion of I/O request signaled
- Particularly useful for **SMP** and **DMA**
- Policy questions such as serialization
- Kernel support needed for true **AIO**

Asynchronous I/O: Posix API

AIO Control Block (ACB):

```
struct aiocb {
 int aio_filedes;      // file descriptor
 int aio_lio_opcode    // operation
 int aio_reqprio       // priority offset
 volatile void *aio_buf; // buffer location
 size_t aio_nbytes;    // length of transfer
 struct sigevent aio_sigevent; //signal No.
 _off64_t aio_offset; //file offset    }
```

Asynchronous I/O: Posix API

Functions in **librt**:

```
int     aio_read (struct aiocb *cb);
int     aio_write ();
int     iol_listio ();
int     aio_error ();
ssize_t aio_return ();
int     aio_fsync ();
int     aio_suspend ();
int     aio_cancel ();
void    aio_init ();
```

Asynchronous I/O: Linux Implementation

- OLD:
  - **glibc** in user space
  - Thread launched for each file descriptor which has pending I/O requests
  - Semantics correct, but expensive
  - Doesn't scale well with many requests
- KAIO: from **SGI**, uses kernel threads, 35 percent performance enhancement
  http://www.oss.sgi.com/projects/kaio

Asynchronous I/O: Linux Implementation

- NEW: Effort led by Ben LaHaise
  - **http://kanga.kvack.org/~blah/aio/**
- Includes libaio
- New **`file_operations`** struct elements **`aio_read(),aio_write(),aio_fsync()`**
- Eventually all I/O will fall under **AIO**
- Earlier implementation in Red Hat 7.3
- Docs at: **http://lse.sourceforge.net/io/aionotex.txt**

IDE Layer Update

- Early 2.5 kernels had a complete rewrite
- Led to stability problems, filesystem corruption
- Led to technical and political upheaval
- 2.4 kernel IDE layer restored, but many incremental improvements made
- Lesson: build new and old at same time

ACL Support (Access Control List)

- More fine-grained permission control:
  - By particular user, group, etc.
- New system calls added for ACL
- Filesystems require ACL support too
  - In **EXT2/3, Reiser, XFS, JFS**
- **http://acl.bestbits.at**

New NTFS Driver

- New Version 2 NTFS filesystem driver (New Technology File System)
- Used in Windows NT, 2000, XP
- Read only (write is very dangerous)
- Reverse engineered
- Mount NTFS volumes on Linux
- **http://linux-ntfs.sourceforge.net/**

Removed Features: Export of `sys_call_table`

- System calls are done by jumping to **`sys_call_table[n]`**
- The table is **exported** to modules
- Thus it is possible to substitute for standard system calls, or introduce new ones.

Removed Features: Export of `sys_call_table`

- Example:

Module loading:

```
save_syscall = sys_call_table[n];
sys_call_table[n] = my_syscall;
```

Module unloading:

```
sys_call_table[n] = save_syscall;
```

Removed Features: Export of `sys_call_table`

- Technical problems (solvable):
  - Unsafe, prone to race conditions
  - Non-portable, different on every platform
  - Security, possibly
- Licensing problems (more basic)
  - Modules should not change heart of kernel
  - Can still put in “stubs” for new calls

Removed Features: End of **Task Queues**

- Task queues were used for deferred processing, *e.g.*, interrupt bottom halves
- Have been replaced by **tasklets**, which are better on SMP systems, cleaner
- Have been deprecated throughout 2.4, so not many instances were left.
- Quick fix was to use **`schedule_task()`**, run under **`keventd`** context; gone now

Removed Features: End of **Task Queues**

- Can convert over to **tasklets,** or:
- New **workqueue** patch (Ingo Molnar)
  - linked list of structures with functions, data
  - executed in process context; sleep ok
  - default workqueue like **`schedule_task`**
  - entries do not block each other

Linux Seminars at OGI

- Linux Kernel 2.6 - New Features I
  - **October 1 @ OGI**
- Linux Kernel 2.6 (3.0?)- New Features II
  - **October 15 @ OGI**
- Linux Network Programming
  - **January 7 @ OGI**
- Online at: http://www.axian.com/pressroom.php
  or http://www.axian.com/learning.php

Upcoming Linux Programming Classes at Axian

- RHD236 (Linux Kernel Internals)
  - **November 4 - 8**
  - **January 27 - 31**
- Linux Kernel Network Programming ***(New)***
  - **December 2 - 6**
- RHD221 (Linux Device Drivers)
  - **January 20 - 24**

Upcoming Linux Programming Classes at OGI

- Linux Kernel Internals
  - **December 16 - 20**
- Linux Device Drivers
  - **November 11 - 15**
  - **January 6 - 10**
- Linux Kernel Network Programming ***(New)***
  - **February 10 - 14**

Upcoming PLUG Meetings

- Thursday, Nov 7, 7PM at PSU, Smith Memorial Center Room 294/296
- Presentation by Zot O'Connor
- **`http://pdxLinux.org`**
- **ALSO:** Monthly PLUG Linux Clinic, Saturday, Oct 19, 1 - 4 pm, at Riverdale HS, 9727 SW Terwilliger Blvd
- **`http://server.riverdale.k12.or.us/~danh`**