Leveraging 2.6 Kernel Features

Andrew Morton
OSDL

2.6 Upward Scalability Improvements

- More and larger disks
- More memory on x86
- More CPUs
- Improved NUMA support
- Improved many-disk I/O performance
- Improved direct-IO support
- Scalable “event polling”

2.6 Downward Scalability Improvements

- Merged support for “nommu” processors
  - M68K, Fujitsu FRV, H8/300, SuperH
- More modular kernel configuration and build system
  - Smaller kernel memory footprint
- More carefully sized kernel data structures
  - Reduced kernel memory usage
- Embedded ISVs will need some kernel expertise
- The “linux-tiny” tree goes much further

Why ISVs Should Test Development Kernels

- We add new features all the time, some of which will be useful to you
- We need to hear of bugs and insufficiencies during or shortly after development
  - Not 18 months later when the code has been stabilized and applications are using the interfaces
- As a strategic investment, test your application(s) on the latest kernel, provide feedback.
- Result will be better for you in 18 months
  - Forward-moving benefit

New Features in Linux Kernel 2.6.x

- **Massive number of new features**
  - **Many could be useful to ISVs**
  - **No centralized listing of features**
  - **New features being added rapidly**
  - **ISVs will need staff who have some familiarity with what the kernel team is doing**
- **The remainder of this presentation will focus on a subset of the new features**

Monitoring and Instrumentation

- Enhanced disk statistics collection
  - SAR and iostat provide more disk traffic info
- Oprofile
  - System-wide symbolic execution profiling. Obligatory!
- Enhanced /proc monitoring
  - /proc/meminfo, /usr/bin/vmstat, /usr/bin/top

CPU scheduler

- Improved performance with many tasks & CPUs
  - O(1) scheduler was backported to 2.4 vendor kernels
  - tuning/sizing decisions should be revisited regularly
- Process <-> CPU binding system calls
  - Allow a process to be bound to a set of one or more CPUs

Synchronization

- FUTEXes (Fast Userspace MuTEXes)
  - A locking primitive which avoids entry into the kernel in the common uncontended case
  - Has been backported into vendor 2.4 kernels
- To leverage the O(1) scheduler and FUTEX features
  - Use a modern distribution (>2.6)
  - Linux pthread API.

Memory management

- Many new tunables in /proc/sys/vm
  - Documented in the kernel source tree
  - Tuning these parameters to a particular workload can yield good results

Filesystems (ext3)

- Much better SMP scalability
- Improved inter- and intra- file layout
- Selecting initial journal size matters
  - Sometimes it matters A LOT
  - Learn to use mke2fs and tune2fs
  - Experiment
- Mounting with the “noatime” option helps
  - Reduced write traffic
  - Reduced buffercache size
- Choose your journalling mode
  - Mounted with “data=writeback” can be faster

Other Filesystems

- Ext2 is fast – consider using it
- Journaling Filesystems
  - XFS is merged in 2.6.
  - reiserfs enjoys some enhancements
  - Leverage web resources for tuning both
- New - reiserfs v4
  - Currently still in development

Direct-IO and AIO-direct-IO

- Present in 2.4 kernels on ext2 and raw disks
  - Some distros disabled it on ext2 due to security issues
- In 2.6 kernel-based distributions
  - Available on numerous filesystems
  - Faster, less memory used, relaxed alignment requirements
- Direct transfer between disk and user memory
  - Very low CPU load, doesn't use caching
- Applications
  - Suitable for large streaming I/O & small random I/O
  - Lack of caching could be a net loss
  - For streaming I/O, should perform large I/O (>0.5MB)

Cache management with posix_fadvise()

- New syscall in 2.6
  - Cf. POSIX 1003.1, 2004 Edition
- Lets applications provide per-file access-pattern hints to the kernel
  - See man page for more info
- Features / options
  - Random access
  - Sequential access
  - Force explicit readahead
  - Explicit pagecache invalidation
    - Use fsync() beforehand if the file is being written

hugetlbfs

- Provides access to the CPU's “large TLB page” feature
  - Access is via a filesystem API and mmap()
- Benefits
  - Databases
  - Applications using large amounts of memory

I/O schedulers

- 2.6 features new I/O schedulers
  - deadline, anticipatory, CFQ
- Selectable via boot option, or at runtime post-2.6.9
- Deadline
  - simple, suitable for seeky (database) loads
- Anticipatory
  - High throughput for general use,
  - Doesn't work well (yet) on SCSI disks with tag-command-queueing
- CFQ
  - attempts to provide similar QoS among different tasks
  - Offers fairness, latency mimimization
  - Still evolving – gaining some AS-like features

Application Hints

- Copying data to/from kernel
  - small buffers often deliver best performance
- Readv() and writev()
  - Got a lot faster in 2.6
  - Now out-perform stdio
- New – epoll()
  - Very scalable select()-like function

Other New Features

- Other new goodies in 2.6
  - POSIX timers
  - NSA security framework
  - crypto API
  - user-mode linux
  - NFS BSD flock() support
  - JFS
  - NFSv4
  - AFS
  - suspend-to-disk
  - IPSec
  - IPVS
  - NAPI
  - partial AIO support
  - Security Enhanced Linux
  - encrypted loopback driver
  - NFS O_DIRECT support
  - NTFS
  - CIFS
  - NFS on TCP
  - POSIX ACLs and Eas
  - bridging firewall
  - IPPC
  - SCTP
  - LVM2
- Lesson: task someone with following the kernel
  - lwn.net is a good way

Conclusion

- The 2.6 kernel offers developers and deployers a rich toolbox
  - Numerous new features and capabilities
  - Benefits to Data Center, Desktop & Embedded
- Distributions based on 2.6 now ubiquitous
  - Features earlier back-ported are native in 2.6
- Tracking 2.6 can be a hefty task
  - Cultivate a kernel expert in your team