The Kernel and the VFS

A Filesystem Engineer's Perspective

Darmstadt, September 1999

sct@redhat.com

Contents

- A brief history of Linux...
- The recent age
- Exciting new stuff
- Even more exciting stuff coming up Real Soon Now

History: The Good Old Days

- Linux users and developers were once the same people.
- Really high hardware requirements: 386, a few KB of memory and a hard disk not much faster than a floppy.
- Filesystems were developed according to:
  - What was cool
  - What was the primary OS
  - What made the world's best development workstation

History: People start getting serious...

- Hey, somebody's using my code in their business!!!
- "Early adopters" included software developers, Universities and ISPs
- Lots of new networking code improvements
- Filesystem was largely good enough for most people
- Primary filesystem requirement was low latency, not scalability

History: Not so very long ago

- A miracle occurs. People start running Oracle on Linux. Everybody else joins in.
- Suddenly, filesystem requirements jump. Dramatically.
- The developers are now coding primarily to other people's needs, not their own
- That's OK, the new problems are Cool And Interesting:
  - Silly levels of performance
  - Massive scalability
  - Enterprise Availability and Manageability

So what is new in 2.2? The "dentry"

- "It's something new"
- This is not Unix-like: no (superblock, inode) indexing
- The VFS passes around pathname components as top-level entities now
  - Networked filesystems **love** this!
- Name lookups from cache can now bypass the individual filesystems entirely
- Lots of hooks for odd name aliasing behaviour

2.2: What else?

- Two other major additions:
- Fine-grained SMP locking
  - Covers the inode cache and IO request layers
- File "read actors"
  - Support sendfile
  - In 2.3, used to support khttpd
- Also lots of new compatibility filesystems and partitioning support

2.3: Hey Everybody, we Fixed the Page Cache!

- (...finally...)
- The buffer cache can be aliased on top of page cache memory
  - Caches the on-disk locations of recent filesystem IOs
  - More and more VFS operations can bypass the underlying filesystem entirely
  - No more double buffering!
  - Separate page-cache SMP locks
  - The filesystems can relinquish control of caching and worry only about *physical* storage.

2.3: The Magic Kiobuf

- The Kiobuf:
- An abstraction between memory ownership and IO on that memory
- The IO layers don't need to care where memory is owned:
  - IO direct from userspace
  - mmaped kernel buffers
  - fd-passing for memory which never visits user space
- Useful for filesystem, framebuffers, networking...

And for our next trick...

- What new things are coming right up?
- Block devices:
  - Shared disk/networked RAID for shared-nothing clustering
  - LVM and Snapshots
- Filesystems:
  - kiobuf-based full async IO and iobuf passing
  - Shared disk and distributed filesystems for shared-everything clustering
  - Faster, bigger: XFS?   GFS? Veritas? cXFS?
  - Maybe we'll even get NFS to work (NFSv4 is coming!)