Linux Filesystems and Storage

Chris Mason
Fusion-io

Linux Growing Up

It wasn't pretty...

Linux – 2000

- 2.2.x kernels
- Mostly SMP capable
- 1024 Process limit
- No journaled filesystems
- No source control system

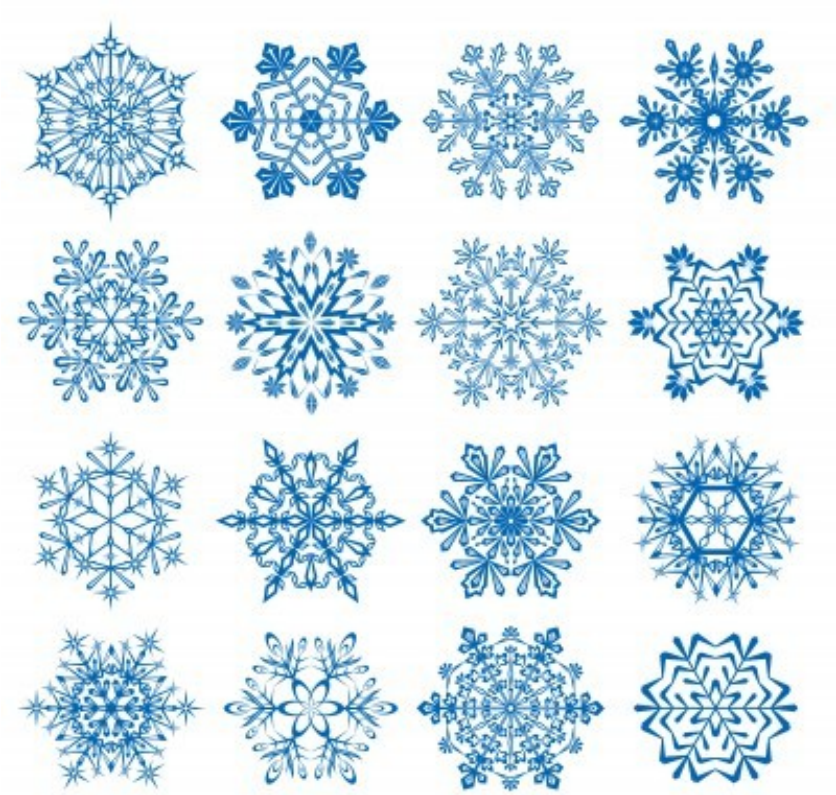
Linux 2.4.x

- Enterprise Ready!
- Start of SMP scalability
- Many journaled filesystems introduced
- Up to 4GB of ram
- Raw IO
- Still no source control system

Enterprise Linux

Early enterprise distributions differentiate with kernel features

Each distro has a large collection of unique changes



Linux 2.6.x

- Even more enterprise ready
- Very long and uncertain 2.5.x development series
- Distributions slowly move their patch piles forward
- Source control at last

2.6.x Scalability

- NUMA
- Block layer
- Page cache
- Networking
- MM subsystem
- AIO / DIO

  …
- The whole kernel development process finally scales

Git

The Kernel Never Stops

- Constant incremental changes
- New APIs and functionality
- Across all subsystems

Since 3.0 (July 2011) ...

- 67,892 commits
- 6.5 commits per hour
- 1.2 million new lines of code
- ~1200 developers per release

Backporting to the Enterprise?

- Traditional enterprise kernels are well behind mainline
- They include large backports of major features
- Some enterprise distributions are rolling out recent kernels
- Important test of the stable kernel series

Mainline – Does it Scale?


```
diff --git a/fs/aio.c b/fs/aio.c
--- a/fs/aio.c
+++ b/fs/aio.c
@@ -1696,7 +1696,6 @@ long do_io_submit(aio_context_t ctx_id, long nr,
     int i = 0;
-    struct blk_plug plug;
 @@ -1716,8 +1715,6 @@ long do_io_submit(aio_context_t ctx_id, long nr,
     kiocb_batch_init(&batch, nr);

-    blk_start_plug(&plug);
-
@@ -1740,7 +1737,6 @@ long do_io_submit(aio_context_t ctx_id, long nr,
     }
-    blk_finish_plug(&plug);
```

Linux Filesystems

Why are there so many?

Where are We Now – Ext4

- Modernized Ext format
- Heavily used at Google (among many others)
- Targeting embedded and large systems
- Some static limitations still present

Where are We Now – XFS

- Significant metadata performance improvements
- Disk format changes will bring metadata checksumming
- Best scalability for large files and large systems

Where are We Now – Btrfs

- Major features not found in other Linux filesystems
- Good overall performance
- Scalability work in progress

Where are We Now – Device Mapper

- Thin provisioning
- Improved snapshot implementation
- SSD front ends under development
- Simplified management tools under development

Where are We Now – CF

- Working with embedded developers to improve filesystem interactions with flash
- Extend flash life time
- Avoid destroying the flash completely
- Improve performance

Where are We Now – Block

- Highest performance storage send bios directly through the device driver
- Linear scalability possible
- Bypasses important features provided by the elevators and SCSI layer

Where are We Now – SCSI

Where are We Now – SCSI

- Strong support for every device type
- Participation in new standards
- 4K Sectors
- UNMAP / TRIM / WRITE SAME
- T10 PI
- Multipathing
- Cgroups (via elevators)

Where are We Now – NFS

- Still the network filesystem
- Revisions introduce new features and complexity
- Interoperability is key to continued success

Futures – Atomic Writes

- Advanced devices can provide true atomic writes down to the media
- New standards and APIs plan to expose this feature to applications and filesystems
- Many difficult implementation details

Futures – Copy Offload

- Block range cloning in storage
- Or – copy offload by the storage
- New token based standard in the works
- Filesystem interactions not fully worked out

Futures – Shingled Drives

- Hybrid storage required to work well with most Linux filesystems

Futures – Hinting

- Data tiers
- Connect blocks likely to be freed at the same time
- IO priorities
- Feedback required to make sure hints are effective

Futures – Flash

- Racing to take advantage of intelligent, seekless storage
- Overlap between flash management and traditional filesystems – how do we avoid doing the same work twice
- Disconnecting locality from performance fundamentally changes how we manage data

Thank You

chris.mason@fusionio.com