Linux Block IO -

present and future

Jens

SUSE Labs

Tokyo 2004/10/15

Agenda

- 2.4 problems
  - scheduling
  - buffer_head
  - scalability
  - API
- 2.6 block layer
  - bio
  - io schedulers
  - requests
  - plugging
  - command transport
- 2.7
- Questions

2.4 IO scheduling

- What?
- elevator_linus
  - One-way scan (C-SCAN variant)
  - Latency
- Data structures
- Run time
  - O(n) sorting
  - O(n) merging
  - O(1) extraction

Struct buffer_head

- 2.4 IO unit
- Big and messy
- sector offset
- Size limitations

Scalability

IBM IO scalability tests (OLS 2002 proceedings)

Kernel / Transfer rate	*MiB/sec*	*CPU idle %*
Linux-2.4	133	21
+ io_request_lock	235	61
+ vary io	240	94
Linux-2.5.17	243	97

Disk microbenchmark: 32 processes, 4K reads of 64KiB size

Scalability components

- Submitting io
  - elevator_linus
  - buffer_head struggle
- io_request_lock
  - Contention
  - Cache line bouncing
- request allocation
  - Low memory pressure
  - Memory waste

New 2.6 io unit

- Goals:
  - Arbitrary amount of data
  - No knowledge of virtual mappings
  - Absolute disk location
    - (hda4, 0)
    - (hda, 1004060)
  - Good stacking
  - Volatile
  - Generic

bio

- New block layer io unit
- Page vector
  - struct bio_vec {page, length, offset}
- bio_alloc(gfp_mask, nr_iovecs)
  - bio_get/put()
- bio_add_page()
  - BIO_MAX_PAGES
- bio->bi_size

IO schedulers

- New defined API
  - merge, add
  - may queue
  - next, remove, requeue
  - init, exit
- 2.6 sports 4
- Selectable at boot time
  - elevator=
- Switchable at runtime
  - /sys/block/<dev>/queue/scheduler
    - noop anticipatory [deadline] cfq

deadline scheduler

- Time based latency control
  - Request service batching
- C-SCAN request delivery
- Seperate read/write queues
- Data structures
  - rb-tree, hash
- Runtime
  - O(log(n)) sorting
  - O(1) merging
  - O(log(n)) next request

Anticipatory scheduler

- What is anticipation?
- Like deadline
  - ’almost’ C-SCAN
  - antic_expire
- Batches reads and writes
- IO contexts
- Performance

CFQ scheduler

- Collision-free variant of SFQ
- Groups on per-process/uid/xxx
- Per-group sorting
- Fairness
  - Reads
  - Writes
- Default in SLES9 and Fedora Core 3

IO Barriers

- Use
- Soft and hard barriers
- SCSI vs IDE
- blkdev_issue_flush()

Plugging

- Concept
- 2.4 vs 2.6
- Was perfect, until...
  - Enter Intel/SGI
- Per-backing device unplugs

Plugging performance

SCSI drive, 16 depth TCQ, ext2
command: diff -urp linux-2.6.5 linux-2.6.8 > /dev/null

Scheduler/Run	***1***	***2***
deadline	151 seconds	152 seconds
as	150 seconds	150 seconds

Plugging enabled (thresh=4,delay=3ms)

Scheduler/Run	***1***	***2***
deadline	154 seconds	154 seconds
as	154 seconds	154 seconds

Plugging disabled

Plugging scalability

Intel system: 32 CPUs, 1000 disks on <unknown>
SGI system: 64 CPUs, 112 disks on 10 qla FC controllers

Results	*Intel*	*SGI*
Linux-2.6.3	40K iops, 95MiB/sec	44K iops
Linux-2.6.3+patch	110K iops, 230MiB/sec	48K iops

SCSI command transport

- 2.4 CD burning sucks
  - ide-scsi
  - CDROM_SEND_PACKET
- SG_IO hijacking
- blk_rq_map_user()
  - bio_map_user()
  - bio_copy_user()

Linux-2.7

- IO priorities
  - CFQ v2
- IO scheduling
  - Online Switching
  - Modular
  - Apples in, apples out
- IO barriers
  - Efficiency
- bsg
  - Full block layer SCSI generic

Questions?