The Linux IO Stack unveiled

Short-Term Substitute Presentation at LinuxTAG 2013
by Thomas Schöbel-Theuer
Agenda

- Where does Linux performance come from?
- How does a cache work?
- Some tuning hints
- Details of IO stack by Werner Fischer & co
Where does Linux performance come from?

- **Cache misses**
- **Cache hits**

### Threads
- Thread 1
- Thread 2
- Thread 3
- .....
- Thread 2001

### VFS layer
- open("/www/customer4711/subdir/x.html", ...)
- read(hugeblock)
- write(postdata)
- close()

### Cache layers
- **VFS layer**
- **dentry cache**
- **inode cache**
- **Page cache / buffer cache**
- **Driver / Hardware**

### File Systems
- ext4
- xfs
- .....

### Performance metrics
- 1 : 100 – 1 : 1000
- << 10.000 ops/s
- < 100 ops/s (avg)
How does a cache work?

Start:

After access to D:

Finally release of D:

aka LRU strategy: keep the HOTTEST items, remove the coldest first
Some tuning hints

- Before tuning anything else, tune your caches!
  - look at /proc/slabinfo
  - don't buy more expensive hardware if you are not sure that you really need it

- Theory of cache tuning: Denning ~1968
  - aka Workingset Theory
  - description at blkreplay.org

![Graph showing Performance vs. Workingset Size]

Thrashing
Decline by factor 100 possible!
The Linux I/O Stack Diagram

version 1.0, 2012-06-20
outlines the Linux I/O stack as of Kernel version 3.3

Applications (Processes)

Applications (Processes) -> VFS

VFS

VFS -> Page Cache

Page Cache

VFS:
- block based FS
  - ext2
  - ext3
  - ext4
  - xfs
  - btrfs
  - ifs
  - iso9660
  - qfs
  - ocfs
  - ...
- Network FS
  - NFS
  - coda
  - smbfs
  - ...
- pseudo FS
  - proc
  - sysfs
  - pipefs
  - futexfs
  - usvfs
  - ...
- special purpose FS
  - tmpfs
  - ramfs
  - devtmpfs

Page Cache

Page Cache

Page Cache -> direct I/O (O_DIRECT)

direct I/O (O_DIRECT)

network

network

optional stackable devices on top of “normal” block devices - work on bios

stackable

stackable

mmap
(anonymous pages)
malloc

read(2)
write(2)
open(2)
stat(2)
chmod(2)

...
IO Stack Details (2) by Werner Fischer & co

Block I/O Layer
optional stackable devices on top of “normal” block devices - work on bios
- mdraid
- device mapper
- drbd
- ...

I/O Scheduler
maps bios to requests
- cfq
- deadline
- noop

BIOS (Block I/O)

SCSI upper layer

request-based device mapper targets
- dm-multipath

hooked in Device Drivers
(hook in similar like stacked devices like mdraid/device mapper do)
- /dev/fio*
- /dev/rssd*
- /dev/iomem-vsl (with module option)
- mtip32xx

/dev/d*
/dev/fio*