# An Introduction to Linux Block I/O

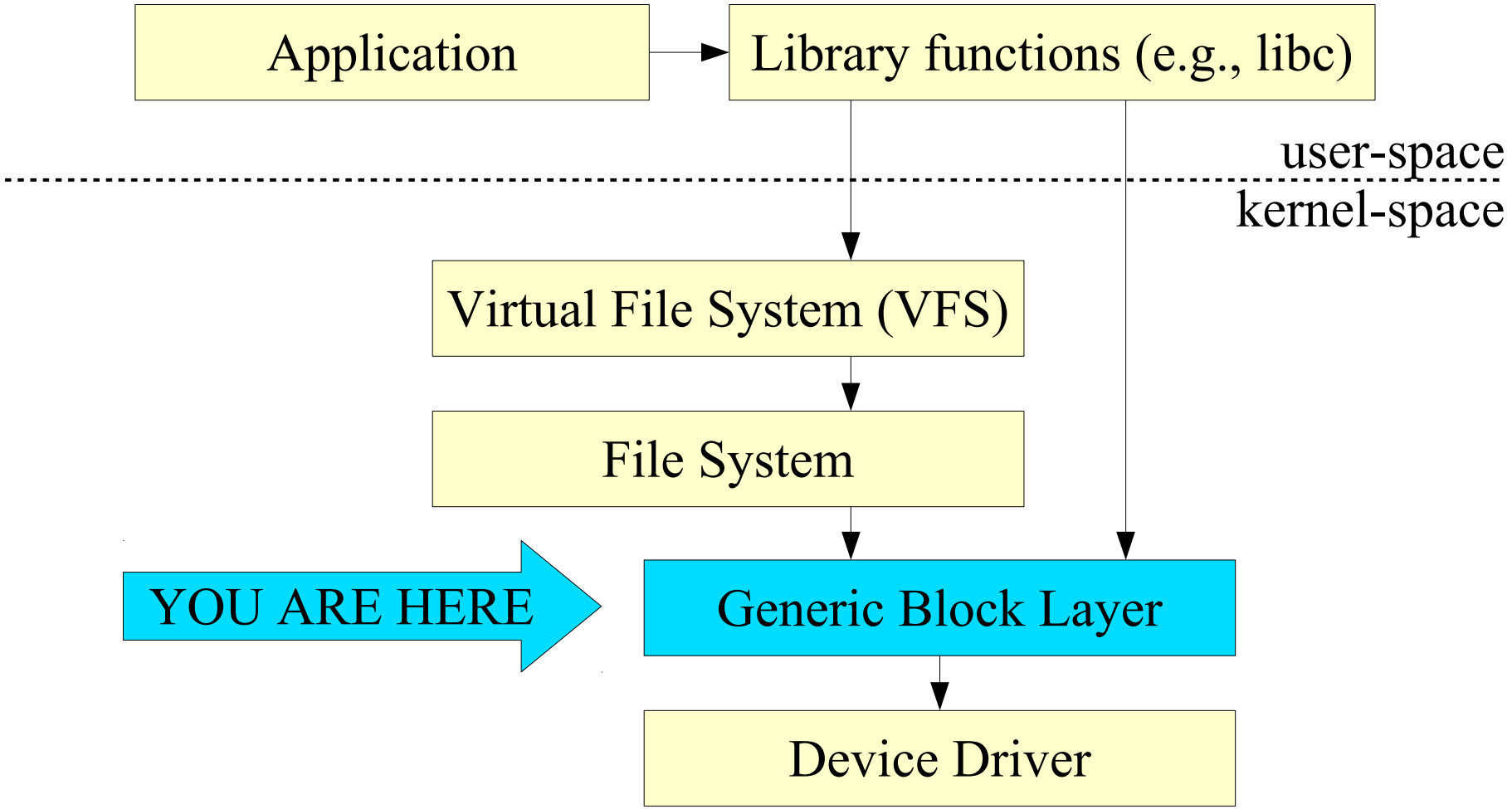
**Avishay Traeger**

IBM Haifa Research Lab
Internal Storage Course

—

October 2012
v1.3

# Where are we?


Application
Library functions (e.g., libc)
user-space
kernel-space
Virtual File System (VFS)
File System
YOU ARE HERE
Generic Block Layer
Device Driver

# Generic Block Layer

- The generic block layer is an abstraction for block devices in the system
- These block devices may be physical or logical (e.g., software RAID, LVM)
  - LVM: group LVs in VGs, R/O or R/W snapshots, striping, mirroring, etc.
- Receives I/O requests in a queue, and is responsible for passing them along to block devices
- The I/O requests are scheduled to be submitted to the devices by an algorithm called an ***I/O scheduler***

# I/O Schedulers

- Goal: maximize block I/O performance
- Schedulers operate on pending requests:
  - **Merge** requests for adjacent sectors
  - **Sort** pending requests based on LBA (*elevator*)
  - **Time** when requests are submitted

- Tricky: must minimize latency, maximize throughput, and avoid starvation
  - e.g., writes-starving-reads: typically synchronous reads starve while the system handles typically asynchronous streaming writes

# Linux I/O Schedulers

- Linux 2.4: Linus Elevator
- Linux 2.6: Deadline, Anticipatory, CFQ, Noop
- Linux 3.0: Deadline, CFQ, Noop

- You can check/set the I/O scheduler for a particular device via:
  /sys/block/sda/queue/scheduler
  (replace “sda” with the name of your device)

```
root@host:~# cat /sys/block/sda/queue/scheduler
noop deadline [cfq]
root@host:~# echo "noop" > /sys/block/sda/queue/scheduler
root@host:~# cat /sys/block/sda/queue/scheduler
[noop] deadline cfq
```

# The Linus Elevator

- Default in Linux 2.4
- Simple elevator that does merging and sorting
- No differentiation between reads and writes
- Starvation problems

# The Deadline I/O Scheduler

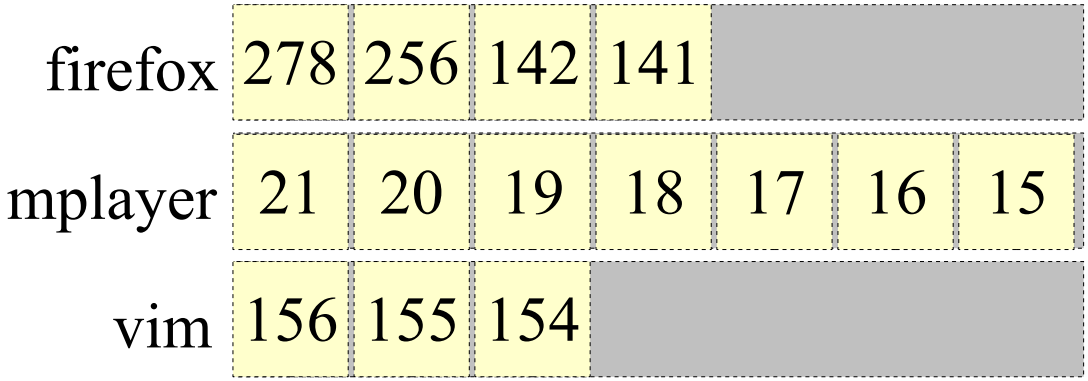
- Goal: fix starvation issue
- Each request gets an expiration time
  - Defaults: 500ms for reads, 5s for writes
- Three queues
  - Sorted queue: sorted similar to Linus Elevator
  - Read FIFO: read requests sorted chronologically
  - Write FIFO: write requests sorted chronologically
- Issue from sorted queue unless a request from the head of read or write FIFO expires

# The Anticipatory I/O Scheduler

- Deadline scheduler problem: System with lots of writes in one area and few reads elsewhere – many seeks, reduced throughput
- Deadline scheduler + anticipation heuristic
- After an expired read is issued, the scheduler waits (default: 6ms) before seeking back and handling sorted requests
- New requests issued to an adjacent area of the disk are immediately handled
- Keeps per-process statistics and has heuristics to improve anticipation accuracy
- Removed from kernel as of 2.6.33 (“mostly a subset of CFQ”)

# The Complete Fair Queuing (CFQ) I/O Scheduler

- Currently the default scheduler
- Per-process sorted queues for sync requests
- Fewer queues for async requests
- Allocates time slices for each queue
- Priorities are taken into account (see ***ionice***)
- May idle if quantum not expired – anticipation



# The Noop I/O scheduler

- Pronounced as “no op”, as in “no operation”
- Performs merging, but no sorting
- Good for devices with truly random access (examples: flash, ramdisk)
- Good for devices that sort requests themselves (examples: storage controller)

# References

- References in this presentation refer to Linux 2.6.35
  - http://lxr.linux.no/#linux+v2.6.35/
- Further reading:
  - Linux Kernel Development (Love): Good for overview – 3rd edition recently published
    - 2nd edition: http://linuxkernel2.atw.hu/ (hopefully posted with the author's permission...)
  - Understanding the Linux Kernel (Bovet & Cesati): Good for reference
  - http://www.ee.ryerson.ca/~courses/coe518/LinuxJournal/elj2004-118-IOschedulers.pdf
  - http://doc.opensuse.org/products/draft/SLES/SLES-tuning_sd_draft/cha.tuning.io.html

# Appendix: Terminology

- **Sector**: Smallest addressable unit, defined by the device (power of 2, usually 512 bytes)
- **Page**: Fixed-length block of main memory that is contiguous in both physical and virtual memory addressing. Smallest unit of data for memory allocation performed by the OS.
- **Block**: Smallest addressable unit, defined by the OS (power of 2, at least sector size, at most page size)
- **Buffer**: Represents a disk block in memory
- **Buffer head**: struct that describes a buffer

# Appendix: The `bio` Structure

- Represents block I/O operations that are in flight as a list of segments
- Segments can be from several non-contiguous buffers (scatter-gather I/O)

- Pending requests are placed on queues
- Each request structure is composed of one or more **`bio`** structures (for consecutive sectors)