Intro to I/O Scheduling

Nick Garvey

Why bother?

During a seek you can...

- Send 1MB of data over a 1 Gbps network
- Read 10MB off of a SSD
- Read 40MB of data from memory
- Send a data center packet 20 times roundtrip
- Send a packet 700 miles
- Do 50 million processor instructions

Numbers Source: http://goo.gl/sBU9pD

Key Concepts

- Maximize global throughput

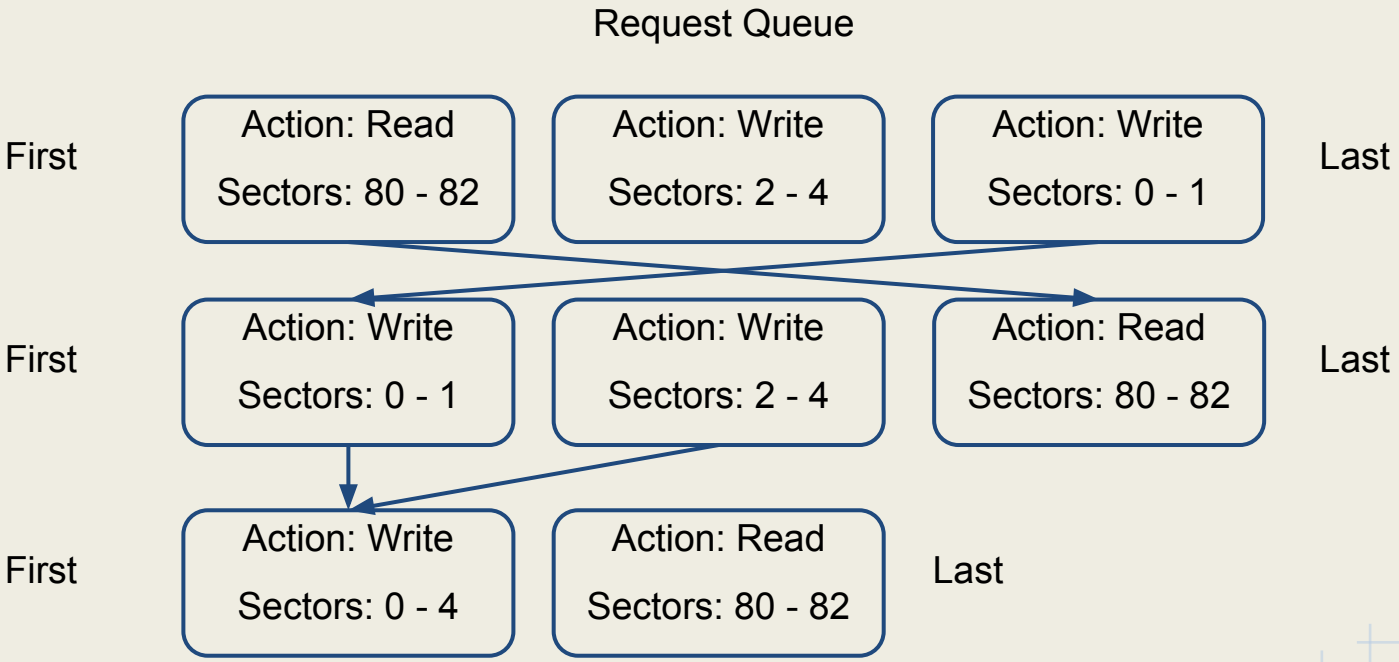
Linus Elevator (Kernel 2.4)

`block/elevator.c`

~~22, 21, 23, 24, 10, 11, 29~~

21, 22, 23, 24, 29, 11, 10

Sort and Merge



Key Concepts

- Maximize global throughput
- **Requests need to complete in some reasonable time frame**
  - **Failure is called "request starvation"**
  - **Trade off with global throughput**
- **Applications tend to wait on reads, don't wait on writes**

Deadline Scheduler

`block/deadline-iosched.c`

- Read queue has deadline of ½ second
- Write queue has deadline of 5 seconds

Key Concepts

- Maximize global throughput
- Requests need to complete in some reasonable time frame
  - Failure is called "request starvation"
  - Trade off with global throughput
- Applications tend to wait on reads, don't wait on writes
- **If sector N is written, it is likely that N+1 will be written**

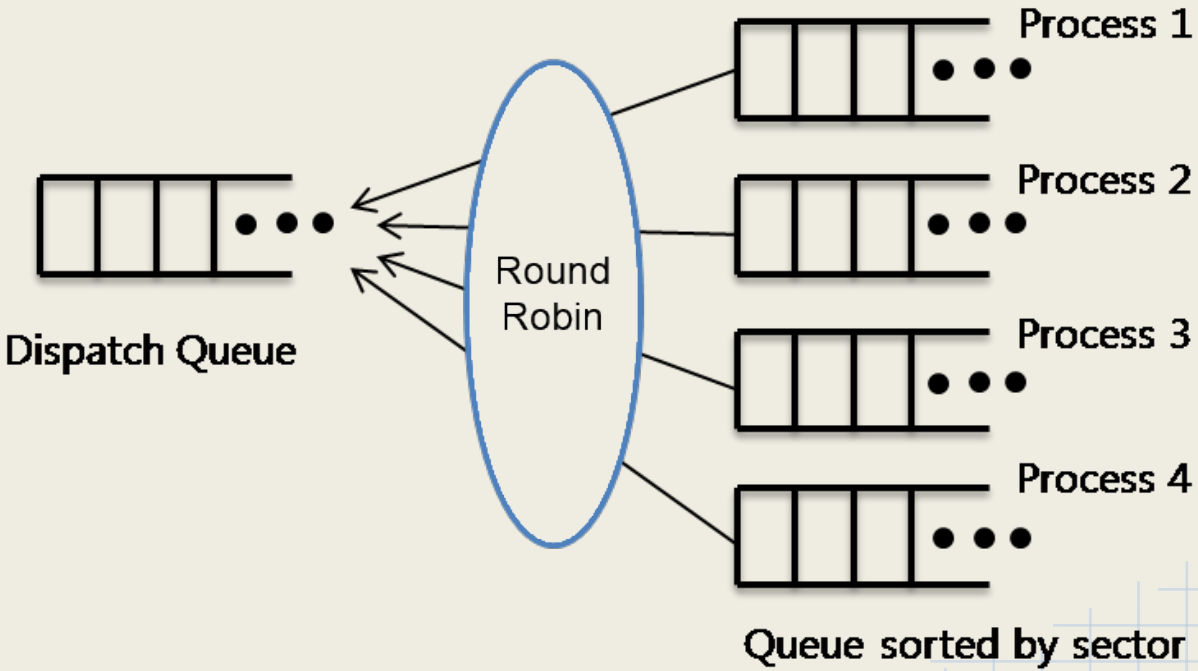
Anticipatory Scheduler

~~`block/as-iosched.c`~~

- Just like deadline scheduler, but waits for more I/O before seeking
- Removed as of 2.6.33 as it is obsoleted by tuned CFQ

Complete Fairness Queueing Scheduler

`block/cfq-iosched.c`



No-Op Scheduler

`block/noop-iosched.c`

- Just a first-in, first-out queue

When to use which?

- CFQ (default) for spinning hard drives
- Deadline for non-spinning drives (e.g. SSDs)
- No-op for virtual block devices (e.g. device-mapper)

```
/tmpfs/kernel-src/block# wc -l *iosched.c
  4651 cfq-iosched.c
   476 deadline-iosched.c
   124 noop-iosched.c
```

Change on the Fly

```
# cat /sys/block/sda/queue/scheduler
noop deadline [cfq]
# echo deadline > /sys/block/sda/queue/scheduler
# cat /sys/block/sda/queue/scheduler
noop [deadline] cfq
```

Summary

- Spinning disk I/O is really slow
- Do I/O sequentially to prevent seeks
- Experiment with deadline I/O on SSDs
- CFQ provides a lot of tunables you should play with