I/O Scheduling
Performance
Mechanisms

• Reordering
  - don't (necessarily) issue the I/Os in arrival order
  - SATF, SSTF, SCAN, CSCAN, etc.

• Merging

• Delaying
  - non-work-conserving schedulers
- synchronous: wait for completion before next issue
- sequential: contiguous sector addresses

- thinktime: time from completion → issue
Anticipation

problem: deceptive idleness
idea: give a process time to issue a new request

\[s_1\] (3ms) 

\[s_2\] (no seek)

\[p_1\] (5ms)

issue 
completion 
idle

issue 
completion 
idle 

\textit{timeout!} 

\textit{reschedule} (seek)
Anticipation

• good performance if:
  \[ t_{\text{antic}} < t_{\text{seek}} \] (average)

• huge performance hit if we get it wrong!

• Iyer and Druschel (SOSP'01)
  
  *Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O*

  - Apache: +29% and +71%
  - AFS: +8%
  - TPC-B: +2—60%
Anticipation

• Linux “as” (Nick Piggin '02)

• When to anticipate?
  - seek distance is “small”
  - thinktime is “small”

• Other considerations
  - nearby requests
  - >1 request issued
  - write issued
  - hardware queueing
  - process disappeared

```c
struct as_io_context {
  /* IO History tracking */
  /* Thinktime */
  unsigned long last_end_request;
  unsigned long ttime_total;
  unsigned long ttime_samples;
  unsigned long ttime_mean;
  /* Layout pattern */
  unsigned int seek_samples;
  sector_t last_request_pos;
  u64 seek_total;
  sector_t seek_mean;
};
```
Plugging

- Merging: combine sector-adjacent requests
- What happens if the queue is empty?

- Plugging: delay new I/O streams
Fair Queueing
Fair Queueing

• Goal: distribute disk resources “fairly” to users and/or processes

• Proportional bandwidth
  – a disk is not a fixed-bandwidth device!
  – available bandwidth depends on access pattern
  – result: poor utilisation

• Proportional time
  – allocate disk in time-slices (c.f. CPU scheduling)
  – e.g. Linux's CFQ
• Completely “Fair” Queueing
  – Jens Axboe, 2003
  – currently at v3

• one queue per thread
  – rb-tree sorted by sector

• service each queue round-robin
• 100ms default time-slice
Reads vs. Writes

• reads: usually synchronous
  – something is waiting for the result
  – latency sensitive!

• writes: usually asynchronous
  – coming from the buffer-cache

• prefer reads
Reads vs. Writes

• solution: handle reads and writes separately
• issue x reads for every write

• problem: lost spatial locality!
Reads vs. Writes

- solution: batching

- as: issue reads for 500ms, writes for 125ms
- deadline: 2 read batches for each write batch
  - 16 requests per batch
- CFQ: lump all writes in a single queue
I/O Priorities

• some threads are more equal than others

• scale time-slice length and scheduling frequency

• CFQ
  - real-time
  - best-effort
  - idle
I/O Priorities

deceptive idleness strikes again!!!
I/O Priorities

• solution: priority “grace period”
• idle x ms before switching to lower priority

• CFQ:
  – wait 100ms before servicing idle class
  – anticipate even for non-sequential I/O

• idle class: you must be root – why?
Hardware
Write-back Buffering

- reduce perceived latency
- overlap disk access with bus traffic
- disk can schedule
- lose your data with well-timed power loss
Write-back Buffering

- can issue many buffered-writes per time-slice
- writes can “escape” the time-slice

- solution: don't use write-back!
- CFQ: limit #writes per slice
- buffer flush
Write-back Buffering

- no completion events
  - when can we try again?
- solution: disable write-back!
- “adversary effect”
TCQ: Tagged Command Queueing

• overlap disk and bus access
• disk can schedule

• completion events!
  - no adversary effect
  - explicit flush unnecessary

• problem: can issue many I/Os in a short period
RAID

• is I/O scheduling useful?

• properties:
  – black box
  – huge queues

• seek-limiting: no gain
• merging: very important!
• fairness
• anticipation: tiny arrays only

• key tradeoff: queue depth
Solid State Drives

... or, why nobody cares about I/O scheduling
SSD

- huge IOPS
- locality (mostly) irrelevant

- scheduling SSDs:
  - ???
From imagination to impact