Fahrrad: An I/O Scheduler for Efficient Guaranteed Disk Performance

Anna Povzner, Tim Kaldewey, Scott Brandt, Richard Golding†, Theodore Wong†, Carlos Maltzahn

UCSC Storage Systems Research Center
†IBM Almaden Research Center

This work was supported in part by NSF Award No. CCF-0621534
Efficient real-time I/O scheduler

Performance guarantees for a mix of applications in a single scheduler

- Arbitrary performance requirements
  - Hard and soft real-time
  Examples:
    - multimedia: timely delivery of data
    - real-time data capture: I/O rate
    - airline reservations: timely response
  - Best-effort
    - Responsiveness expectations
Goals

- **Timeliness guarantees**
  - Full range of hard real-time, soft real-time, and best-effort applications
  - Arbitrary reservation granularities

- **Good performance**
  - Efficient use of disk resources

- **Good isolation**
  - Performance should not depend on behavior of other I/O streams
Fahrrad scheduling model

- Utilization reservations
  - Disk time per period

- RAD gives utilization guarantees
  - Supports full range of application requirements

- Adapting RAD to disk scheduling
  - Allocation: disk time per period
  - Dispatching: based loosely on EDF
  - Ordering: head scheduling
Basic Fahrrad scheduler
Utilization reservation guarantees only

- Assumes I/Os are available in the beginning of each period
- I/Os are non-preemptible
- I/O times are not known a priori

\[\begin{align*}
A &: u = 20\% \quad p = 250 \\
B &: u = 50\% \quad p = 500 \\
C &: u = 10\% \quad p = 500 \\
BE &: u = 20\% \quad p = 500 \\
\end{align*}\]

Approach:
- Assume worst-case I/O time first
- Track actual I/O time
- Dispatch more I/Os for streams whose usage < reservation
Basic Fahrrad scheduler
Implementation

A
\[ u = 20\% \quad p = 250 \]
B
\[ u = 50\% \quad p = 500 \]
C
\[ u = 10\% \quad p = 500 \]
BE
\[ u = 20\% \]

\[
\begin{array}{cccc}
500 & 375 & 250 & 125 \\
300 & 250 & 150 & 50 \\
200 & 150 & 100 & 50 \\
125 & 100 & 50 & 25 \\
\end{array}
\]

EDF and micro-deadline
Shift micro-deadlines
\[ \Delta = (\text{WCRT-a})/u \]
Actual I/O time
EDF and micro-deadline < deadline
WCRT/utilization
Disk
Dispatcher
Fahrrad optimized for performance

Order of requests matters

- Allow efficient request reordering
- But should not violate deadlines

Horizon = earliest deadline

Shift micro-deadlines
$\Delta = (\text{WCRT-a})/u$

Dispatch all requests up to horizon

Disk scheduling set (DSS)
- largest set of requests that can be executed in any order

Actual I/O time
Isolation for throughput guarantees

- Ordering that extends DSS as soon as possible
  - Extend stream deadline as soon as it has used its budget
  - Send I/Os from streams with earliest deadline first
  - Move to next horizon as soon as earliest deadline met

- Account for seeks from other streams
  - At least 1 seek to and away from stream with deadline = horizon
Dealing with unqueued requests

Hold onto reservations as long as possible to provide good guarantees
- Hold empty slots in the DSS
- Expire slots if requests do not arrive
- Donate expired disk time to other streams

Utilization is guaranteed if each I/O arrives no later than its micro-release time.
Performance enhancements

- Empty slots of bottleneck streams prevent early deadline extension ⇒ isolation problems
- Increase contiguity of requests in the DSS of non-bottleneck streams
  - Trade slots in DSS with slots outside the DSS
  - Get max number of requests from one stream

A
u = 20%
p = 250

B
u = 50%
p = 500

C
u = 10%
p = 1s

BE
u = 20%
p = 1s

DSS

Disk
Performance and isolation

Utilization and throughput of I/O streams as period of stream 4 changes

- All streams are sequential and always have I/Os queued up

Workload:

- **SRT 1**: 20%, period 2 sec, sequential, many outstanding I/Os
- **SRT 2**: 20%, period 125 ms – 1 sec, sequential, many outstanding I/Os
Performance and isolation (2)

Utilization and throughput of I/O streams as period of HRT stream changes

Workload:
- SRT 1: 20%, period 2 sec, sequential, I/Os are always available
- SRT 2: 20%, period 125 ms – 1 sec, sequential
- SRT 3: 20%, period 125 ms – 1 sec, sequential
- HRT: random
- BE: random
Comparison with Linux

Workload:

media 1: 400 sequential I/Os per second, 20% utilization reservation

media 2: 800 sequential I/Os per second, 40% utilization reservation

transaction: short bursts of random I/Os at random times, 30% reservation

background: random
Existing schedulers with QoS guarantees

- Some degree of isolation, but no real-time guarantees
  - Examples: YFQ, Hierarchical Disk Sharing, Zygaria

- Particular classes of real-time guarantees
  - Example: Clockwise targets multimedia I/O

- Multiple classes of real-time and best-effort workloads
  - Hierarchical schedulers: Cello, MARS

- Optimizing performance while meeting real-time guarantees
  - Examples: SCAN-EDF, Globally Seek-Optimizing Rescheduling Scheme (GSR)

- Most similar system to Fahrrad: Scheduling framework in DROPS
Conclusions

- Correct real-time scheduling for a combination of hard and soft real-time I/O streams within a single scheduler
- Ordering mechanism allows to obtain good performance
- Early deadline extension, scheduling bottleneck streams first, and slot swapping help to achieve throughput isolation