Efficient Guaranteed Disk I/O Performance Management

Anna Povzner, Scott Brandt, Carlos Maltzahn
Richard Golding, Theodore M. Wong
Darren Sawyer

SRL Research Symposium
October 19, 2010
Many workloads have specific performance requirements
- Throughput and latency
- Timeliness: hard, soft, best-effort

Multiple workloads share the system → interference
Performance guarantees for a mix of workloads

- Applications reserve and the system guarantees:
  - Throughput and latency
  - Timeliness: hard, full range of soft, and best-effort
Guaranteeing Disk I/O is Challenging

- Mechanical nature of hard drives
  - Requests are non-preemptable, stateful, non-deterministic
  - Response times vary by orders of magnitude

- Good guarantees \textit{and} good performance
  - Allow request re-ordering without violating guarantees

- Meaningful guarantees \textit{need} perfect isolation
  - Interference between workloads destroys sequentiality
  - Hard, workload-independent performance guarantees
Contributions

- Fahrrad real-time disk I/O scheduler
  - Hard guarantees on reservations and high I/O performance

- Hard (and soft) isolation guarantees
  - Guardion virtual disks – illusion of solo access to the disk

- Hard and soft throughput and latency guarantees

- Predictable performance for multiple disks

- Conclusions
Hard performance guarantees

- **Traditional:** Throughput reservations
  - Based on worst-case response time assumptions
  - >99.99% of bandwidth is *not* reservable

- **Fahrrad:** Disk time utilization reservation
  - Based on actual response times
  - ~100% of bandwidth is reservable

**Reservation:**
- *disk time utilization*
- *period* (reservation granularity)

- 20% of disk time every 250 ms
- 30% every minute
- 50% of disk time every second
- Time on the disk device
Fahrrad real-time disk I/O scheduler

**Hard utilization guarantees**
- Assume worst-case time first
- Track actual I/O times
- Dispatch more I/Os from streams whose usage < reservations

**High I/O performance**
- Allow efficient request reordering
- But do not violate any deadlines

```
A
u = 20%  p = 250
B
u = 50%  p = 500
C
u = 10%  p = 500
BE
u = 20%  p = 500
```

Disk Scheduling Set (DSS)

Update usage

Actual I/O time

Anna Povzner                         SRL Research Symposium
Guarantees for varying arrival patterns

Hold reservations as long as possible

- Hold empty slots in the DSS
- Donate slots if requests don't arrive
Performance isolation

- Re-ordering is limited by shortest deadline
- Minimize the number of inter-stream seeks
  - Maximize the number of requests from one stream in DSS

Good isolation... but not guaranteed
- Some seeks are necessary → interference
- Trade off latency for isolation? [e.g. Argon]
  → Guarantee isolation for arbitrary throughput and latency reservations
Hard isolation guarantees

- Account for all extra (inter-stream) seeks and other interference

- Reserve overhead utilization for time to perform extra seeks

\[ \text{Utilization}_i = \text{Disk Share}_i + \text{Overhead utilization}_i \]

Ensure that extra seeks do not affect any guarantees

- Charging model: bill streams responsible for inter-stream seeking from correct reservation (actual or overhead)
Extra (inter-stream) seeks

- Caused by latency reservations
  - At most **two** extra seeks per stream per period
  - Charge stream responsible for extra seek from overhead

- Caused by stream's unqueued requests
  - Depends on stream's behavior
  - Charge stream responsible for extra seeks from reservation

Summary: Hard performance guarantees:

\[
\text{Overhead utilization}_i = \frac{\text{WCRT}}{p_i} + 2*\frac{\text{WCRT}}{p_i} + \frac{\text{WCRT}}{p_i}
\]

Less than worst-case for softer guarantees...
Virtual Disk

- **Reserved** and **isolated** share of the disk time

  Workload performance depends only on: its workload behavior, its reserved share, base device performance

Guardion Virtual Disks

Amount of data transferred:

\[
D(x\%, t) = D(100\%, x\% \cdot t)
\]

Data transferred by virtual disk with share \(x\%\) during time \(t\) = Data transferred using disk alone during time \(x\% \cdot t\)
Hard isolation guarantees = “virtual disk”

- No isolation – performance is lower than standalone
- Guaranteed isolation – performance is close to standalone

![Graphs showing data transfer vs period of virtual disk 3 for different scenarios.]

Anna Povzner  SRL Research Symposium
The cost of hard isolation guarantees

- Hard isolation guarantees need greater reserved overhead
- Overhead is proportional to period (latency reservation)

No isolation

<table>
<thead>
<tr>
<th>Reserved: each virtual disk</th>
<th>Total: VD3 (10% share, 250ms-2s)</th>
<th>Total: VD2 (10% share, 500ms)</th>
<th>Total: VD1 (10% share, 2s)</th>
</tr>
</thead>
</table>

Hard isolation guarantees

<table>
<thead>
<tr>
<th>Reserved: each virtual disk</th>
<th>Total: VD3 (10% share, 250ms-2s)</th>
<th>Total: VD2 (10% share, 500ms)</th>
<th>Total: VD1 (10% share, 2s)</th>
</tr>
</thead>
</table>
Throughput and latency guarantees

- Applications make performance reservations via **broker**
  - Throughput and/or latency
  - Timeliness: hard, soft, best-effort

- Fahrrad guarantees utilization

→ guarantees throughput and latency
Translating requirements into reservations

- Fahrrad reservation incorporates
  - throughput/latency
  - Timeliness (confidence level)
  - Workload behavior

- Incorporating workload behavior
  - Eliminates worst-case assumptions
  - More reservable throughput

Broker

- Throughput and/or latency
- Timeliness requirements
- Expected I/O behavior

- Disk time utilization
- Period
Guaranteeing throughput

- N requests per period $\rightarrow$ disk time to service N requests

- Incorporating I/O behavior
  - read/write ratio, I/O size, % of random, arrival pattern [shriver97]

- We use simple device measurements
  - No detailed knowledge of the device

$$Utilization = U_{\text{random, read}} + U_{\text{random, write}} + U_{\text{sequential, read}} + U_{\text{sequential, write}}$$
Random throughput

Throughput with a given confidence level

\[ U_{\text{random}} = \text{expected}\_\text{io}\_\text{time}(\text{confidence}, N_{\text{random}}) \times N_{\text{random}} \]

- Random service times are very variable
- Service times for sets of request are less variable
Sequential throughput

- Most service times are in order of hundreds on microseconds
  - Occasional head/track switches (2-3 ms)
- Account for bad layout!
  - Logical sequentiality is not always sequential
  - Outliers can significantly affect sequential throughput
We guarantee throughput reservations

- Workloads received reserved throughput with requested confidence level

A sequential workload with N IOPS reservation and random background

Broker

IOPS & confidence $\rightarrow$ utilization

Fahrrad

99.5% confidence
95% confidence
90% confidence

Utilization reservation (%)

Throughput reservation [IOPS]

Throughput [IOPS]
We guarantee latency reservations

- Reservation granularity bounds latency: period = latency/2
- Reserve utilization to maintain arrival rate

Periodic semi-sequential stream that shares storage with random background stream.
Guarantees for multiple disks

- Fahrrad: performance guarantees for one disk

Multiple disks
→ how to make reservations?
→ data layout?

- Fahrrad can be used in any system managed with reservations
  - Stonehenge: allocates storage capacity and storage performance
  - Ceph: distributes data pseudo-randomly over all devices [Wu et al]
  - Ongoing work: end-to-end performance management in Ceph
Guarantees without resource reservations

- Complex and dynamic data placement policies, data paths
  - E.g. FAB, Data ONTAP GX

Decentralized management of disk resources
- Throughput and latency targets
- Efficient utilization of system resources
- High performance

NetApp clustered storage system
Multi-layered management of disk resources

- **throughput limiting**
- **deadline assignment**

- Workloads specify performance targets
  - Throughput and latency

- Upper-layer control mechanisms
  - Throughput limiting
  - Deadline assignment based on throughput and latency targets

- Low-level disk schedulers
  - Meet individual request deadlines
  - Maintain high efficiency
  - limited re-ordering
  - Arbitrary order of request deadlines
  - Overloaded disk resources
Horizon real-time disk I/O scheduler

- Manage I/O in terms of disk time
  - Dispatch based on expected I/O times
  - Track actual I/O times
  - Re-order based on “slack” before earliest deadline (horizon)

- Estimating I/O service times
  - Histories of recent service times
  - Sequential/random, reads/writes

- Implemented as storage device driver in NetApp's Data ONTAP 8

More details:
A. Povzner, D. Sawyer, S. Brandt "Horizon: efficient deadline-driven disk I/O management for distributed storage systems," HPDC 2010
Meeting performance targets for multiple disks

40% random, 40 in flight
1000 IOPS target

media, 40 IOs / 100ms
target: 400 IOPS, 80 ms latency

bursty: 4 IOs / 40 ms
40ms latency target

bursty: 8 IOs / 40 ms
40ms latency target

random background

Media ................ 96% of IOs met latency target

Bursty workloads… ≥ 92% of IOs met latency target

Anna Povzner
SRL Research Symposium
Conclusions

- Hard guarantees and high I/O performance
- Guaranteed isolation → virtual disks
- Arbitrary soft and hard throughput and latency guarantees
- Applying Fahrrad principles in real word
  - Locally managing disk resources for global performance targets
Thank you!

Questions?