Anticipatory Disk Scheduling

Sitaram Iyer  
Peter Druschel  
Rice University

Disk schedulers
- Reorder available disk requests for
  - performance by seek optimization,
  - proportional resource allocation, etc.
- Any policy needs multiple outstanding requests to make good decisions!

With enough requests...

E.g., Throughput = 21 MB/s (IBM Deskstar disk)

With synchronous I/O...

E.g., Throughput = 5 MB/s

Deceptive idleness

Process A is about to issue next request.

but

Scheduler hastily assumes that process A has no further requests!

Proportional scheduler

Allocate disk service in say 1:2 ratio:

Deceptive idleness causes 1:1 allocation:
Prefetch
Overlaps computation with I/O.
Side-effect:
avoids deceptive idleness!

• Application-driven
• Kernel-driven

Prefetch
• Application driven - e.g. aio_read()
  - Application need to know their future
  - Cumbersome programming model
  - Existing apps need re-writing
  - May be less efficient than mmap
  - aio_read() optional

• Kernel driven
  - Less capable of knowing the future
  - Access patterns difficult to predict, even with locality
  - Cost of misprediction can be high
  - Medium files too small to trigger sequential access detection

Anticipatory scheduling
Key idea: Sometimes wait for process whose request was last serviced.

Keeps disk idle for short intervals. But with informed decisions, this:
• Improves throughput
• Achieves desired proportions

Cost-benefit analysis
Balance expected benefits of waiting against cost of keeping disk idle.

Tradeoffs sensitive to scheduling policy e.g.,
1. seek optimizing scheduler
2. proportional scheduler

Statistics
For each process, measure:
1. Expected median and 95percentile thinktime

   ![Graph showing thinktime distribution]

2. Expected positioning time

   ![Graph showing positioning time]
Cost-benefit analysis for seek optimizing scheduler

\[ \text{best} := \text{best available request chosen by scheduler} \]
\[ \text{next} := \text{expected forthcoming request from process whose request was last serviced} \]

 Benefit = \[\text{best.positioning_time} - \text{next.positioning_time}\]

 Cost = \[\text{next.median.thinktime}\]

 Waiting.duration = \[(\text{Benefit} > \text{Cost}) \? \text{next.95percentile.thinktime} : 0\]

Proportional scheduler

Costs and benefits are different.

* e.g., proportional scheduler:
  - Wait for process whose request was last serviced,
    - 1. if it has received less than its allocation, and
    - 2. if it has thinktime below a threshold (e.g., 3ms)

 Waiting.duration = \[\text{next.95percentile.thinktime}\]

Experiments

- FreeBSD-4.3 patch + kernel module (1500 lines of C code)
- 7200 rpm IDE disk (IBM Deskstar)
- Also in the paper: 15000 rpm SCSI disk (Seagate Cheetah)

Microbenchmark

<table>
<thead>
<tr>
<th>Throughput (MB/s)</th>
<th>Sequential</th>
<th>Alternate</th>
<th>Random within file</th>
</tr>
</thead>
<tbody>
<tr>
<td>no prefetch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prefetch</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
</tr>
<tr>
<td>Anticipatory</td>
</tr>
</tbody>
</table>

Real workloads

What’s the impact on real applications and benchmarks?

- Andrew benchmark
- Apache web server (large working set)
- Database benchmark

Overall 8% performance improvement

Andrew filesystem benchmark

<table>
<thead>
<tr>
<th>Execution time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
</tr>
<tr>
<td>Anticipatory</td>
</tr>
</tbody>
</table>

Lower is better
**Apache web server**
- CS.Berkeley trace
- Large working set
- 48 web clients

**Database benchmark**
- MySQL DB
- Two clients
- One or two databases on same disk

**GnuLD**
- Concurrent: 68% execution time reduction

**Intelligent adversary**

**Proportional scheduler**

**Conclusion**
- Anticipatory scheduling:
  - overcomes deceptive idleness
  - achieves significant performance improvement on real applications
  - achieves desired proportions
  - and is easy to implement!
Anticipatory Disk Scheduling

Sitaram Iyer  
Peter Druschel

http://www.cs.rice.edu/~ssiyer/r/antsched/