QUANTITATIVE ESTIMATION OF THE PERFORMANCE DELAY WITH PROPAGATION EFFECTS IN DISK POWER SAVINGS

FENG YAN, XENIA MOUNTROUIDOU, ALMA RISKA, EVGENIA SMIRINI

1 Computer Science Department
College of William and Mary

2 EMC Corporation
**WHY DISKS?**

- **Storage**: main power consumption component
- **Disk level**: portability and scalability, blackbox
- **Disks are underutilized**: power savings potential
## Disk Power Saving Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Power Saving</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arm unloaded Full rotation speed</td>
<td>48% of operational</td>
<td>0.5 sec</td>
</tr>
<tr>
<td>Arm unloaded Reduced rotation speed</td>
<td>60% of operational</td>
<td>1 sec</td>
</tr>
<tr>
<td>Arm unloaded No rotation Electronics on</td>
<td>70% of operational</td>
<td>8 sec</td>
</tr>
<tr>
<td>Disk fully spin down</td>
<td>95% of operational</td>
<td>20 sec</td>
</tr>
</tbody>
</table>

- Hitachi Global Storage Technologies, “Power and acoustics management”
- Seagate Technology, “Constellation ES: High capacity storage designed for seamless enterprise integration”
How to do the power savings?

What is a good scheduling strategy?

User Performance Guarantees

Max Power Saving Amount
Penalty Time: from power saving to active

How to schedule **transparently**?

*Performance Degradation* = Extra Delay / Original RT

≤ Pre-defined User **Performance Target**
User Performance Guarantees

Max Power Saving Amount

Power Saving Amount = \frac{\text{Time in Power Saving Mode}}{\text{Total Idle Time}}
STATE OF THE ART SCHEDULING

Aggressive Scheduling

UTIL-guided Scheduling

Fix-wait Scheduling

Our Scheduling
Our Scheduling

- **Idle time ≤ I**
  - Power Savings: ×
  - Performance Degrade: ×

- **I < Idle time < I + T**
  - Power Savings: ✓
  - Performance Degrade: ✓

- **Idle time ≥ I + T**
  - Power Savings: ✓
  - Performance Degrade: ×
OUR SCHEDULING

Cumulative Distribution Histogram

Average Power Savings:
\[ \sum (\text{Prob.} \times \text{Time}) \]

Average Extra Delay:
\[ \sum (\text{Prob.} \times \text{Delay}) \]

Average Penalty due to Power Savings
If $I < i_1 < I + T$

$1 \leq W_1 \leq P$

1st delay
If $l < i_1 < l + T$

$1 \leq W_1 \leq P$

$\text{1st delay}$

If $W_1 > i_2,$

$W_2 = W_1 - i_2 \leq P - 1$

$\text{2nd delay}$
If $1 < i_1 < 1 + T$

$1 \leq W_1 \leq P$

If $W_1 > i_2$,

$W_2 = W_1 - i_2 \leq P - 1$

$\vdots$

If $(i_2 + i_3 + \ldots + i_k) < W_1 < (i_2 + i_3 + \ldots + i_k + i(k+1))$,

$W_k = W_1 - i_2 - i_3 - \ldots - i_k \leq P - k + 1$ (1 ≤ k ≤ P)
**Prob(P) = \(CDH(I+T-P) - CDH(I)\)**

**Only at first delay**

**Prob(P-1) = \(CDH(I+T-P+1) - CDH(I+T-P) + Prob(P) \times CDH(1)\)**

**Case1: at first delay**

**Case2: at second delay**
Average Extra Delay:
\[ \sum (\text{Prob.} \times \text{Delay}) \]
Our Scheduling

User Performance Guarantees

Max Power Saving Amount
### General Trace Description

<table>
<thead>
<tr>
<th>Trace</th>
<th>UTIL (%)</th>
<th>Idle Length</th>
<th>Mean Arrival Rate</th>
<th>Mean Service Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE1</td>
<td>5.6</td>
<td>192.6, 8.4</td>
<td>0.0089</td>
<td>0.1596</td>
</tr>
<tr>
<td>CODE2</td>
<td>0.7</td>
<td>1681.6, 2.3</td>
<td>0.0013</td>
<td>0.1859</td>
</tr>
<tr>
<td>FILE1</td>
<td>1.7</td>
<td>767.5, 2.3</td>
<td>0.0033</td>
<td>0.1938</td>
</tr>
<tr>
<td>FILE2</td>
<td>0.7</td>
<td>2000.2, 2.3</td>
<td>0.0011</td>
<td>0.1596</td>
</tr>
</tbody>
</table>

**Low UTIL challenge necessity of CDH**
**CDH of idle period length**

EVALUATION  Workload Analysis

![Graph showing the cumulative distribution function (CDF) of idle period lengths with a log scale. The graph indicates the probability of an idle period exceeding a certain duration. A vertical line marks 500 ms, highlighting a significant probability increase at this threshold.]
Orders of magnitude more degradation!

10 times more Power Savings!
Performance Delay Estimation with Delay Propagation Effects

Verified with enterprise trace driven simulations

User Performance Guarantees

Max Power Saving Amount
FUTURE WORK

- Explore clustering idleness case
  - e.g. autocorrelation in consecutive idle periods

- Cross correlation with busy periods
  - Better estimation and scheduling

- Implementation
  - Linux kernel + IO driver
  - Benchmark
THANK YOU!

Questions?