LECTURE 5: WEAR LEVELING AND PERFORMANCE
Summary

- WA reduces with amount of spare
- LRU GC: simple
- Greedy GC: optimal
- Greedy approaches LRU with large $N_p$
- Another advantage of LRU: Wear leveling
The Fundamental Property of Flash Storage

**Theorem:**
For a device with \#E_units=n, after a workload of nt uniform writes, the probability that there exists an E_unit with more than \( \log n \) dirty P_units tends to 0 as \( n \to \infty \).

**Balls and Bins:**
n bins, M balls
1/n: probability of ball falling in bin i
Balls-and-Bins Max Occupancy

**Implication:**
- No block with “very many” dirty pages.
- $t$ on average, $\log n$ max.
- Greedy not fundamentally better than random choice

**Proof:**

$Pr[M \text{ balls or more in bin } i] < ?$

$Pr[M \text{ balls or more in any bin}] < ?$

$Pr[\log n \text{ balls or more in any bin}] \rightarrow 0$
Beyond Uniform Workloads

1. **Hot/Cold** logical addresses

- Can separate **hot** and **cold** to two independent mapping layers with same over-provisioning factor
- Same A as with uniform
- Can do better. How?
p-Local Workloads

2. Time-locality with parameter $p$

$$p$$

$$1-p$$

Recently accessed Lpages

Other Lpages

uniform

uniform
p-Local Workloads

2. Aging parameter $h$

User writes: 1, 2, ..., $h$, $h+1$, ...

Recently accessed Lpages

Other Lpages
Write Amplification with p- Locality

Graph showing the relationship between p (x-axis) and A (y-axis) with a line marked as greedy.
Wear Leveling

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
</tr>
</tbody>
</table>
Erase Count

$N_p=1$, $T=U=1$

$E_{\text{unit}}$ 1

$E_{\text{unit}}$ erase count limit: $K$

Total number of writes before end of life = $1 + K$

$H \ H \ H \ \ldots \ H$

$K$ updates
Uneven Wear

$N_p = 1$, $T = U = 2$

E_unit 1

E_unit 2

E_unit erase count limit: $K$

Total number of writes before end of life = $2 + K$

$C_1 \ H_2 \ H_2 \ H_2 \ldots \ H_2$

K times $\rightarrow$ used only K erases out of the total 2K
Leveling the Wear

\[ N_p=1, \ T=U=2 \]

Can we get to 2\(K+2\) writes? \textit{Almost}...

Example logical write sequence:
\[ C \ H \ H \ H \ ... \ H \]

2\(K\) times

Physical write sequence:
\[ C_1 \ H_2 \ H_2 \ H_2 \ ... \ H_2 \ C_2 \ H_1 \ H_1 \ H_1 \ ... \ H_1 \]

\[ \text{total writes} = 2K+1 \]
Wear-Leveling Algorithm

Theorem:
For $T=U=n$, $N_p=1$, any sequence of $nK$ write requests can be fulfilled.

Algorithm:

(1) Copy logical unit $L$ into $E_{\text{unit } i}$ if
   $$\text{wear}(i) + \text{remaining\_writes}(L) = K-1$$

Proof idea:

(*) Every $L$ is copied at most once. → overhead up to $n$
(*) All unused wear can be claimed by (1) operations
On-Line Wear Leveling

• Previous algorithm required knowledge of full access sequence (off line).

• What if wear-leveling is required for on-line accesses?
### On-Line Wear-Leveling Algorithm

**Algorithm OL1:**

<table>
<thead>
<tr>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>...</th>
<th>En</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>L2</td>
<td>L3</td>
<td></td>
<td>Ln</td>
</tr>
</tbody>
</table>

1. Assign Li to Ei, for all i
2. Li request $\rightarrow$ write in Ei

**Theorem:** Algorithm OL1 is optimal

**Proof:**

Host can always issue a request to $\text{argmax}[\text{wear}(E_i)]$

---

**On-line wear leveling not possible without over-provisioning**
**Algorithm OL2:**

<table>
<thead>
<tr>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>...</th>
<th>Em</th>
<th>Em+1</th>
<th>...</th>
<th>En</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>L2</td>
<td>L3</td>
<td>...</td>
<td>Lm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Assign $L_i$ to $E_i$, for all $i$
2. $L_i$ request $\rightarrow$ write to unused $E_j$ with lowest wear

Algorithm OL2 guarantees $(n-m+1)K$ updates (OL1 special case $n=m$)

**Theorem:** Algorithm OL2 is optimal
Proof:

Adversarial host:
(1) Fix a set of $S=n-m+1$ physical E_units
(2) Issue write requests only to Li’s in $S$

Idea: There is always at least one Li allocated in $S$, hence $\sum_{j \in S} \text{wear}(E_j)$ grows by at least 1 every write

Total updates at most $|S|K=(n-m+1)K$
Performance of Algorithms

Let $A$ be an algorithm with inputs taken from $\chi$. Denote the performance of $A$ on input $x \in \chi$ by

$$Q[A(x)]$$

**Example:** wear lifetime

$$Q[A(x)] = \max [t: \forall j \in \{1, \ldots, n\} wear_t(j) \leq K]$$

$$x = [Req_1, Req_2, \ldots, Req_t, \ldots]$$
Worst-Case Performance

1) Worst-case performance

\[ Q_{wc}[A] = \min_{x \in \chi} Q[A(x)] \]

Every \( x \in \chi \) gives \( Q[A(x)] \geq Q_{wc}[A] \).
2) Average-case performance

Let $x \sim D(\chi)$, $D$ is a probability distribution on the inputs.

$$Q_{av}[A] = E_D[Q[A(x)]]$$

Often $D = U$ is the uniform distribution.
1) Offline performance

\[ y = A(x) \]

A emits outputs only when the full \( x \) is given.

1) Online performance

A emits outputs / commits actions after every \( x_i \).

\[
\begin{align*}
&x_1 \\
&x_1, x_2 \\
&x_1, x_2, ..., x_i \\
&\rightarrow A \\
&\rightarrow y_1 \\
&\rightarrow y_2 \\
&\rightarrow y_i
\end{align*}
\]
Deterministic vs. Probabilistic

1) Deterministic A

\[ x \rightarrow A \rightarrow y \]

A is known to all.

1) Probabilistic A (also called randomized)

\[ x \rightarrow A \rightarrow y \]

\[ c \uparrow \]

A is known to all. Input c (random coin flips) is private.
1) Adversarial input

B knows A and chooses x

\[ Q_{adv}[A] = \min_{x = B(A)} Q[A(x)] \]

If A is deterministic:

\[ Q_{adv}[A] = Q_{wc}[A] \]

If A is randomized:

\[ Q_{adv}[A] \geq Q_{wc}[A] \]

\[ Q_{adv}[A] = \min_{x \in \chi} E_{c} Q[A(x, c)] \quad Q_{wc}[A] = \min_{x \in \chi} \min_{c \in C} Q[A(x, c)] \]
Deterministic vs. Randomized Wear Leveling

\[ N_p=1, \, T=U=N, \, \text{wear\_limit}=K \]

**Alg. Det.**

- Write in **fixed** \( E_{\text{unit}} \)
- 1 erase per \( L_{\text{write}} \)

**Alg. Rand.**

- Write in **random** \( E_{\text{unit}} \)
- 2 erases per \( L_{\text{write}} \) (with probability \( 1-1/U \))

<table>
<thead>
<tr>
<th>Worst case</th>
<th>( K )</th>
<th>( = )</th>
<th>( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial</td>
<td>( K )</td>
<td>(&lt;)</td>
<td>#ball-pairs, s.t. ( \leq K ) in all bins</td>
</tr>
<tr>
<td>Sequential</td>
<td>( NK )</td>
<td>( &gt;)</td>
<td>( NK/2 )</td>
</tr>
</tbody>
</table>
Performance Evaluation

#balls s.t. \( \leq K \) in all bins

**Q:**
Given \( n \) bins and bound \( K \) on number of balls, how many balls \( m \) can throw?

**A:**
Poisson approximation:

\[
\lim_{n \to \infty} \Pr(X_n = k) = p(k) \triangleq \frac{e^{-t} t^k}{k!}
\]

\[
0.001 > \sum_{k=K+1}^{\infty} p(k)
\]

What is \( t \)?

\[ t = \frac{m}{n} \text{ (expectation). Find } t \text{ and substitute } m = tn. \]
Other Topics in Data Placement

• Caching/pre-fetching
  – Move data between fast and slow media to maximize R/W performance

• Compression and data reduction
  – Can improve access and wear performance
  – Challenges to mapping layer

• Workload detection and prediction
  – Tailor placement to workload features

• Security and access control
  – Data privacy, access privileges