Technion

Electronics
Computers
Communications

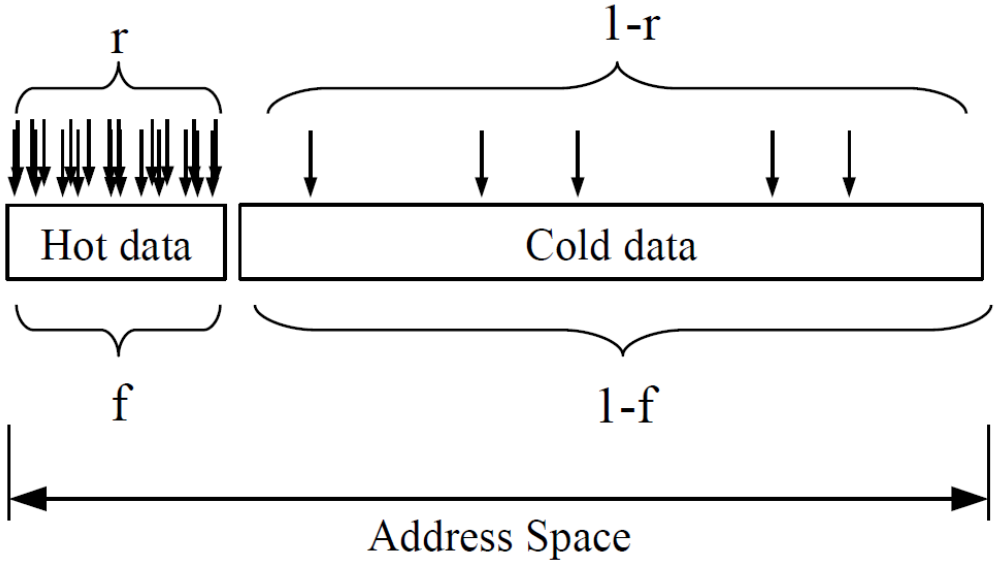
Information in Storage Devices

049063 – EE Department, Technion

LECTURE 5: WEAR LEVELING

Beyond Uniform Workloads

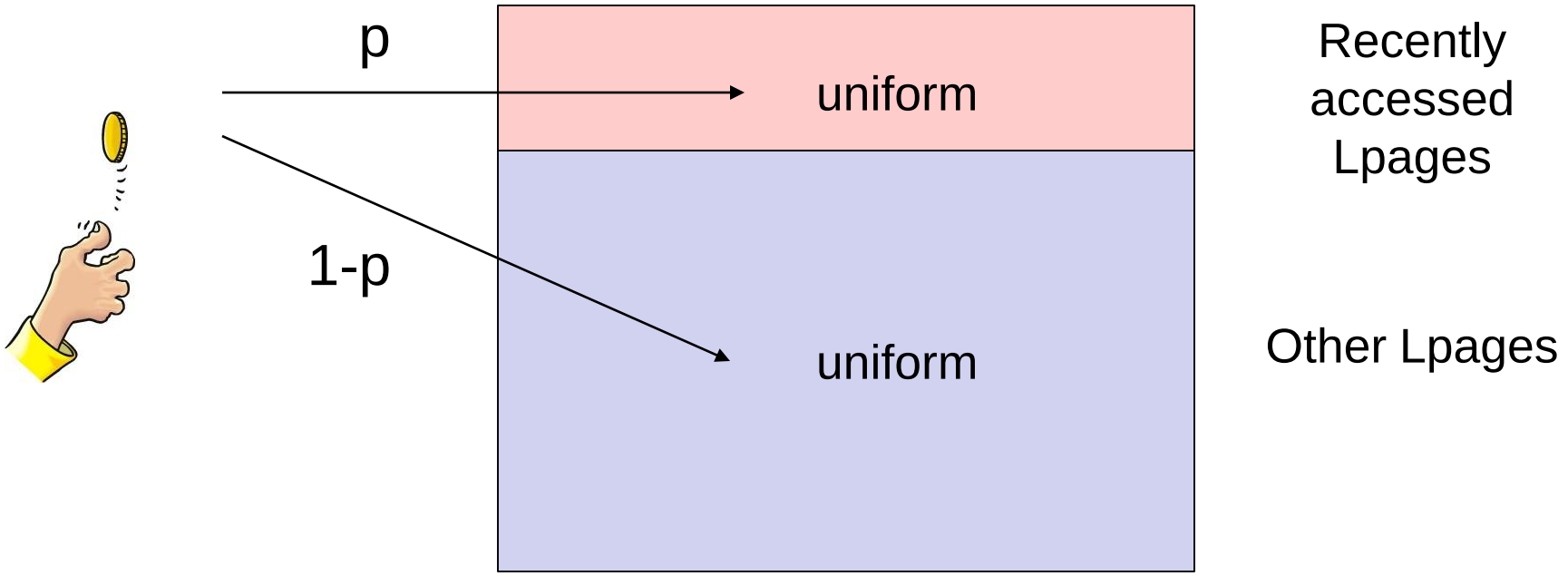
1. Hot/Cold logical addresses



- 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

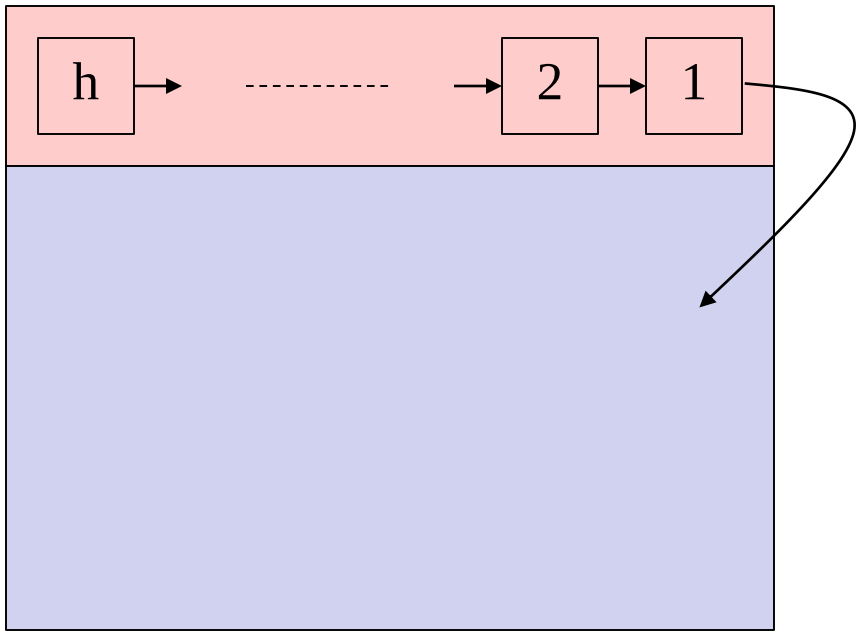
2. Time-locality with parameter p



p-Local Workloads

2. Aging parameter h

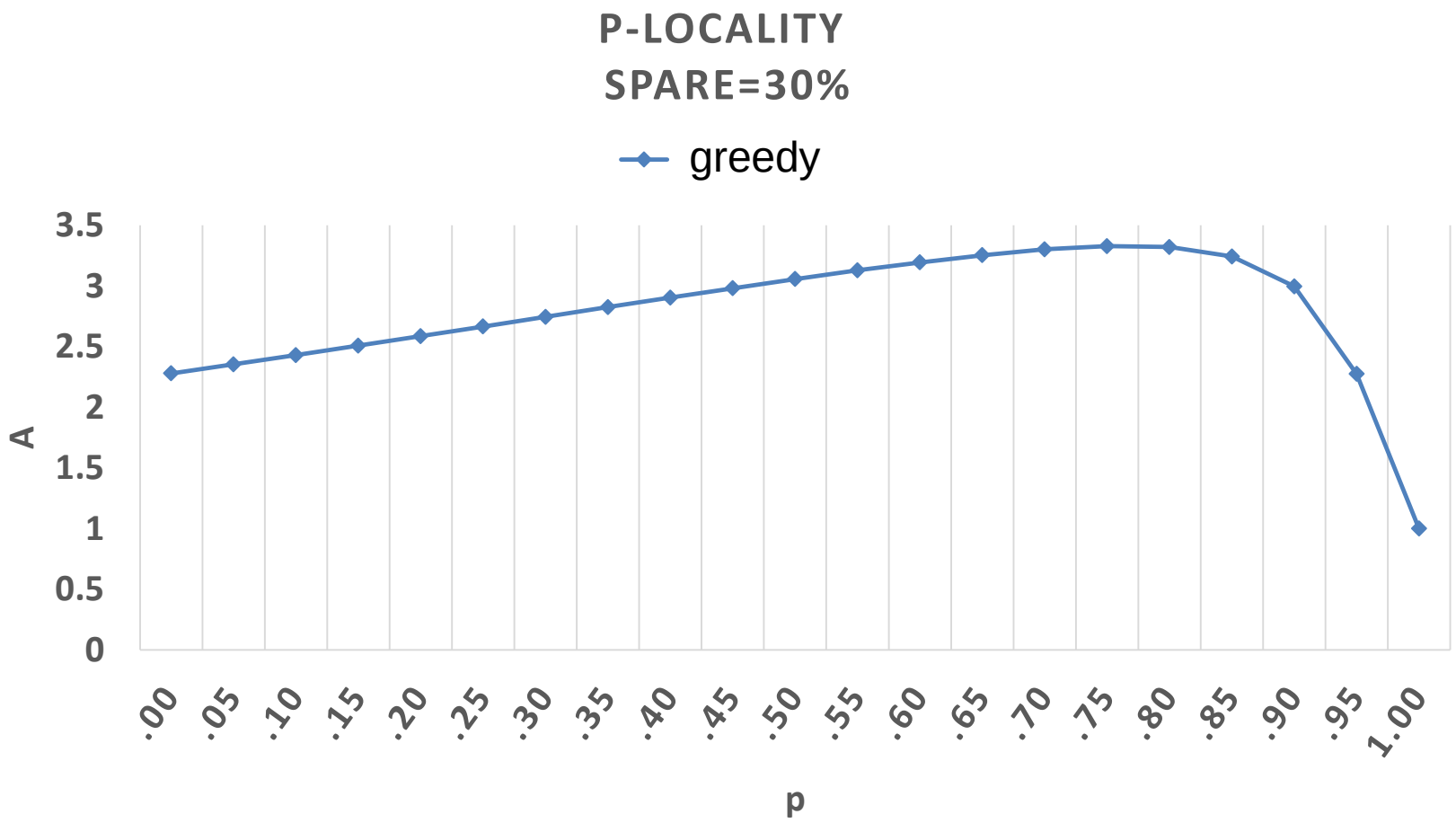
User writes:
1,2,…,h,h+1,…



Recently accessed Lpages

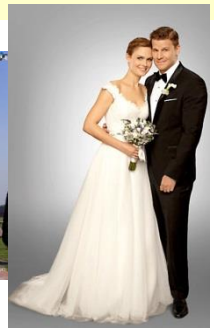
Other Lpages

Write Amplification with p-Locality

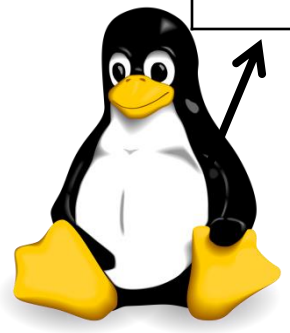


Wear Leveling





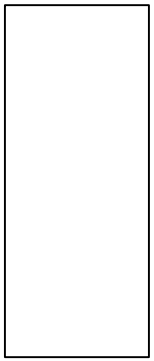
0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15



Erase Count

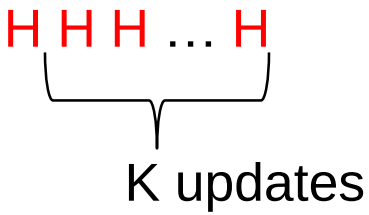
N_p=1, T=U=1

E_unit 1



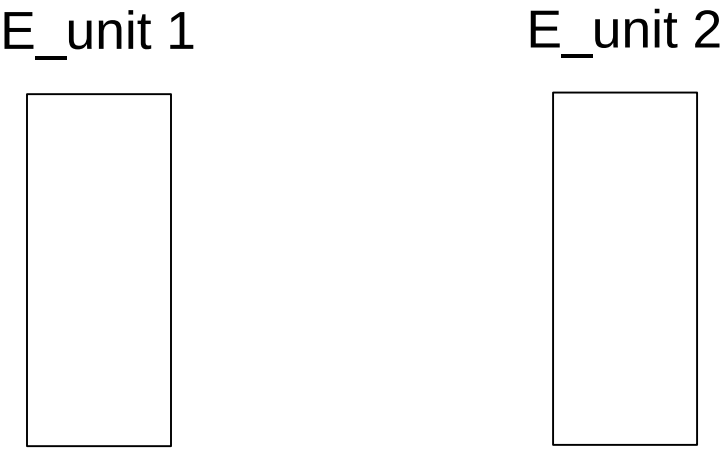
E_unit erase count limit: K

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



Uneven Wear

N_p=1, T=U=2



E_unit erase count limit: K

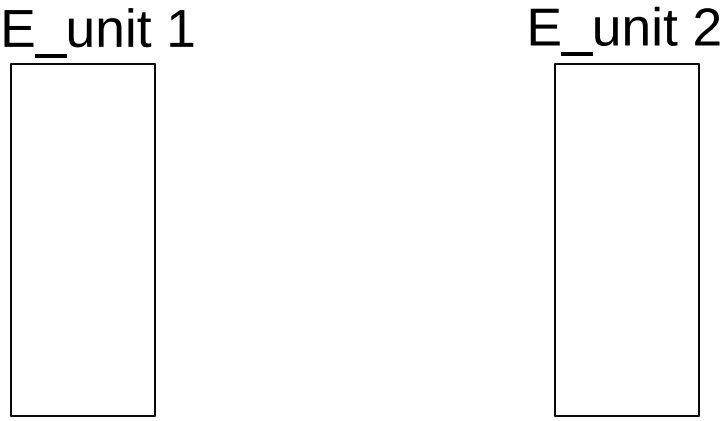
Total number of writes before end of life = 2 + K

C1 H2 H2 H2 … H2

K times ⟶ used only K erases out of the total 2K

Leveling the Wear

N_p=1, T=U=2



Can we get to 2K+2 writes? **Almost…**

Example logical write sequence:

C H H H … H

2K times

Physical write sequence:

C1 H2 H2 H2 … H2 C2 H1 H1 H1 … H1 ⟶ total writes = 2K+1

K-1 copy K

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_unit i if

wear(i) + **remaining_writes**(L) = K-1

Proof idea:

(*) Every L is copied at most once. → overhead up to n

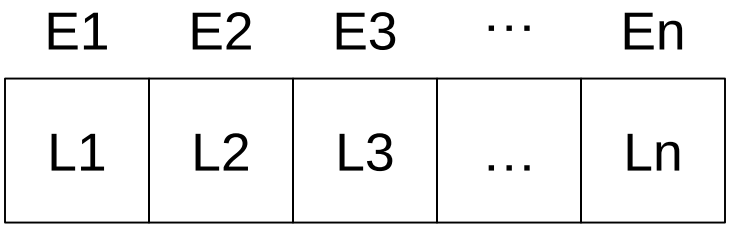
(*) All unused wear can be claimed by (1) operations

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:



(1) Assign Li to Ei, for all i
(2) Li request → write in Ei

Trivial algorithm!
guarantees only K
updates for any n.

Theorem: Algorithm OL1 is optimal

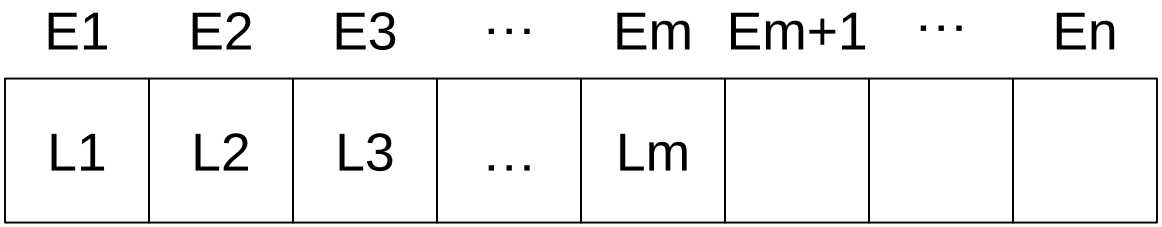
Proof:

Host can always issue a request to argmax[wear(Ei)]

On-line wear leveling **not** possible without over-provisioning

On-Line + Over-Provisioning

Algorithm OL2:



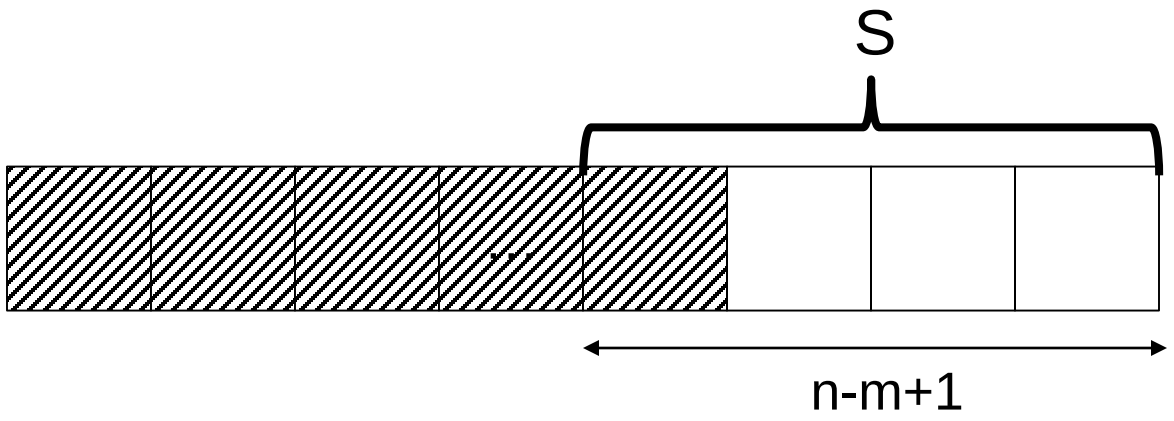
(1) Assign Li to Ei, for all i
(2) Li request → write to unused Ej with lowest wear

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

Theorem: Algorithm OL2 is optimal

Optimality Proof

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 $\sum_{j \in S} wear(E_j)$ grows by at least 1 every write

↓

Total updates at most |S|K=(n-m+1)K

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