Flash Talk

Counting Problems in Flash Storage Design

Bongki Moon

Department of Computer Science
University of Arizona
Tucson, AZ 85721, U.S.A.
bkmoon@cs.arizona.edu

Counting Problems

- **Counting is**
  - **One of the fundamental problems, found in most applications**
  - **The study of approximate counting has a long history in computer science**
- **Goal of this talk is**
  - **Bring attention to the issues, provide a new angle to the design of flash storage systems**

Counting Problems in Flash

- **Wear Leveling**
  - Keep track of block erase counts
  - For a 128 GB SSD with 1 million blocks
    - A (sub)word for each block requires 2~4 MB RAM
- **Garbage Collection**
  - Keep track of # of valid pages in blocks
    - For example, FMAX and FAST
  - For a 128 GB SSD with 1 million blocks
    - A byte for each block requires 1 MB RAM

Counting Problems in Flash

- **Hot / Cold Data Separation**
  - Keep track of update counts for all pages(?)
  - For a 128 GB SSD with 64 million pages
    - A (sub)word for each page requires 128~256 MB RAM
- **Counting Bloom Filter [Hsieh TOST'06]**
  - Approximate update frequencies by storing relative frequencies in a fixed number of fixed-length bit-vectors
  - Aging (divide-by-2) occurs when any counter overflows
  - Accuracy will be degraded seriously at the presence of a few very hot items, because aging kicks in too often
    - Almost impossible to separate cold data pages from warm ones

Approx Counting

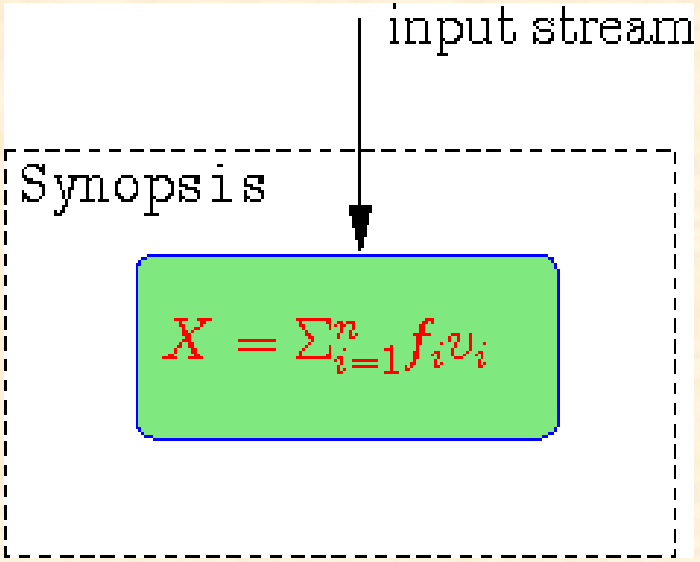
- Numerous approaches
  - Unbiased vs biased
  - Memory use: linear vs. logarithmic
  - …
- Log Count
- Lossy Couting
- AMS Sketch
- …

Counting Heavy Hitters

- **Manku & Motwani [VLDB'02]**
  - ***Biased*** **estimation of frequencies for heavy hitters**
    - **For all heavy hitters ($f_{true} > \tau = sN$), without false negatives, the estimated frequency f_{est} is found**
    - **Always underestimates but with margin $\leq \varepsilon N$, *that is*, $f_{est} \leq f_{true} \leq f_{est} + \varepsilon N$**
    - ***N* = # of objects seen so far, $0 < \varepsilon << s < 1$**
- **Sticky Sampling and Lossy Counting**
  - **LC deems superior to SS, and uses *O(log N)* memory in the worst case**

AMS Sketch

- **Alon, Matias and Szegdy [STOC'96]**
  - **Randomized linear projection of a frequency vector $F = (f_1, \ldots, f_n)$ for a set of n objects in stream S ($|\text{Dom}(S)| = n$)**
- **Randomized linear projection**
  - **f_i denotes the frequency of i in S**
  - **v_i is a four-wise independent binary random variable**
    - ***v_i = +1 or -1 with equal probability, i.e., $Pr(v_i = +1) = Pr(v_i = -1) = ½$***
    - ***Pr(4-tuple of v_i = 4-tuple of {-1,+1}) = 1/16***



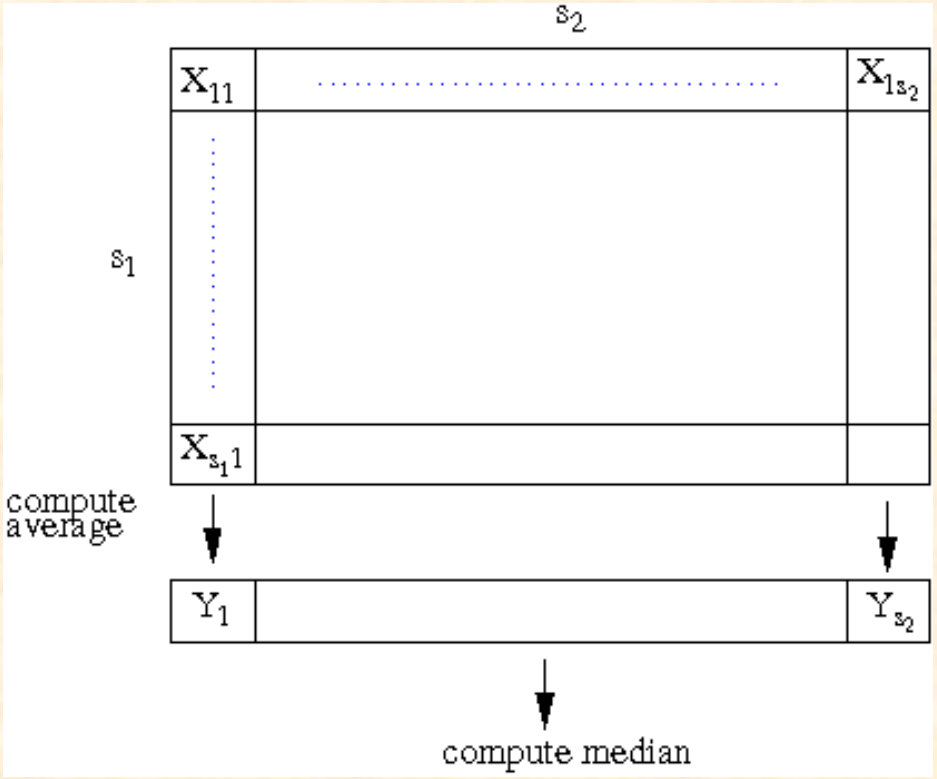
Unbiased Estimation by AMS

- To compute the sketch X
  - Initialize X=0
  - Add v_i to X each time i appears in S
    - $X = \sum f_i v_i$
- To estimate f_q (the frequency of q)
  - $E(X \cdot v_q) = E((f_1 v_1 + \ldots + f_n v_n) v_q) = E(f_q v_q^2) = f_q$
    - Linearity of expectation
    - $E(v_i^2) = 1$, because v_i^2 is either $(-1)^2$ or $(+1)^2$
    - $E(v_i v_j) = 0$ for $i \neq j$, because $Pr(v_i v_j = +1) = Pr(v_i v_j = -1) = ½$
  - Thus, $X \cdot v_q$ is an unbiased estimator of f_q

How Accurate?

- Not so accurate
  - $Var(X \cdot v_q) \leq |SJ(S)| = \sum f_i^2$
- To improve the accuracy
  - Maintain $s_1 \times s_2$ independent and identically distributed instances of X, say X_{ij}
  - Compute s_2 (column-wise) averages of $X_{ij} \cdot v_q$
    - $Y_j = avg(X_{1j} \cdot v_q + \ldots + X_{s1,j} \cdot v_q)/s_1$
  - Then, take the median of them as an estimate

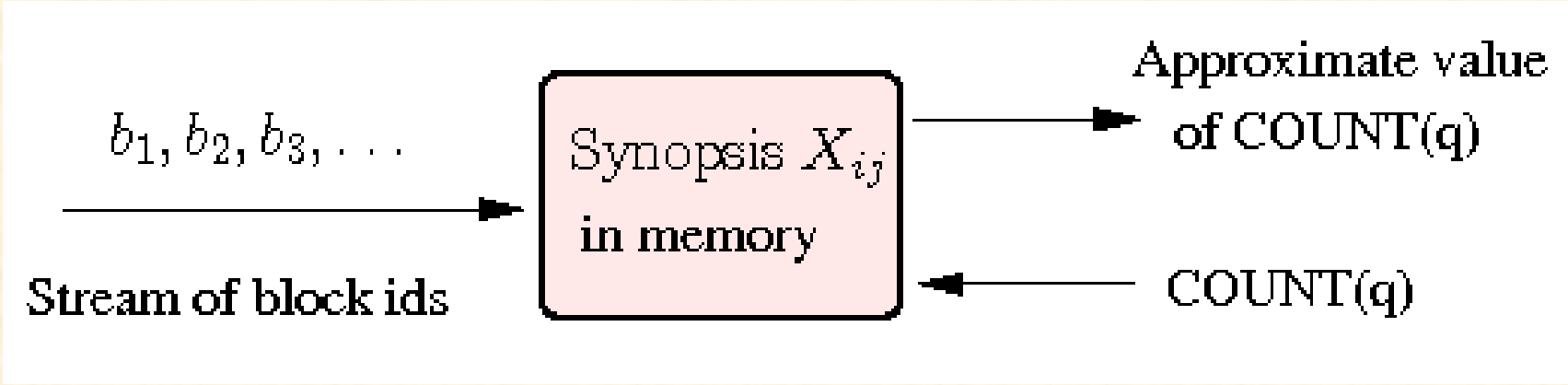
Memory Usage by AMS



- s_1 controls accuracy, and s_2 controls confidence
  - The values of s_1 and s_2 are determined by target error bound and confidence level
  - Still, the total memory use is *O(log|S| + log n)*

Approx Counting for Wear Leveling

- **AMS sketch adopted for wear leveling**
  - **Store synopsis of block erase counts instead of actual counts, consuming much less memory**
  - **Patent KR 10-0817204 [Min & Moon, March 2008]**



Problem Solved?

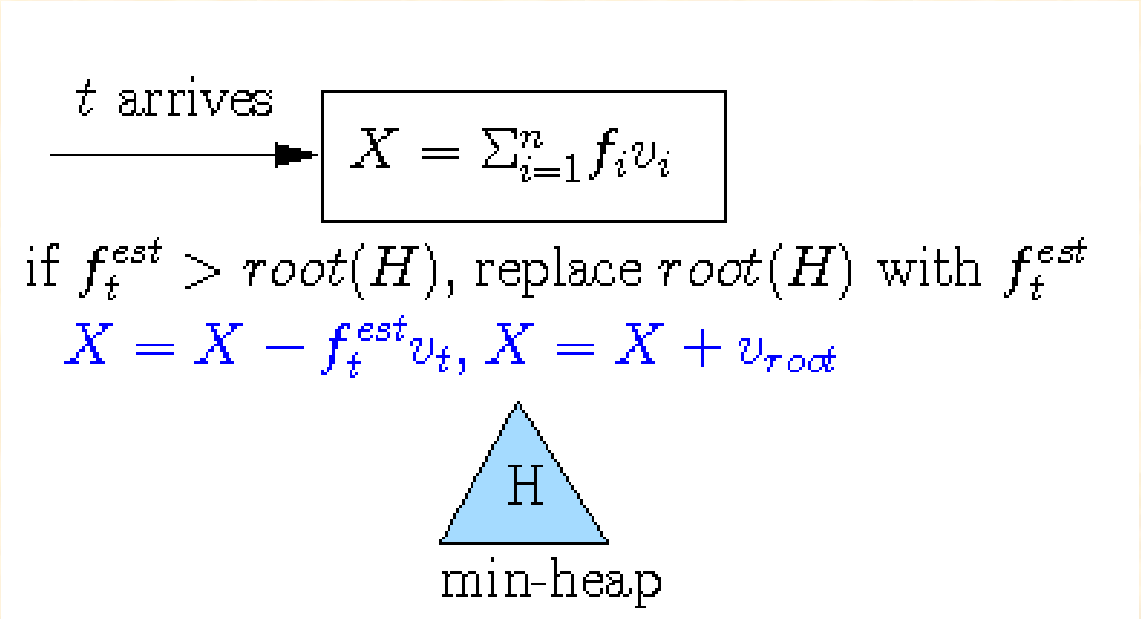
- ***Far from it!***
  - AMS returns an approximate count for a query
  - But it keeps neither a hot list nor a cold list
- **Dynamic wear leveling**
  - Select a youngest block from a free list
  - AMS may work well if the free list is short
    - By submitting each block in the free list as a query
- **Static wear leveling**
  - Need find blocks storing cold data
  - AMS will be extremely slow!
    - By submitting ALL blocks as a query

SW Leveler

- **Chang et al. [DAC 2007]**
  - **Use a bit vector (BET) to mark blocks erased during a certain period; BET is just a set of Boolean flags**
  - **Space-efficient**
    - **For an SSD of 128GB (1 M blocks), the size of BET is just 128KB/2^k (for $k \geq 0$)**
- **However, serious drawbacks**
  - **Any block *<u>un-erased during the period</u>* can be randomly chosen as a cold data block; very inaccurate**
  - ***Memoryless*, because BET is reset between the periods**
    - **No long-term information on block erasures is accumulated**

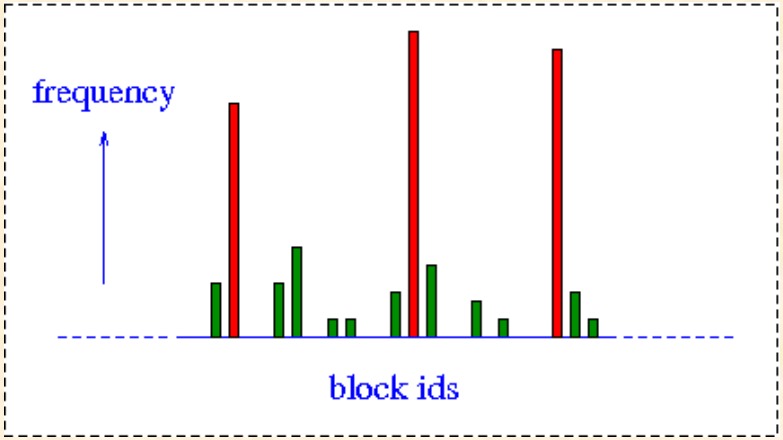
Hot Separation by AMS

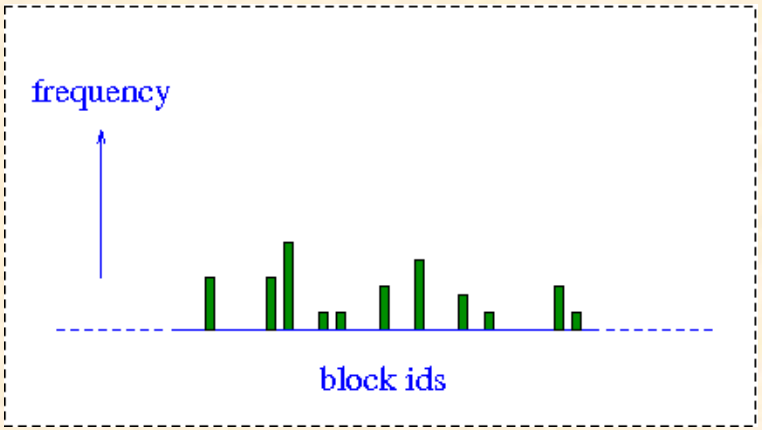
- **Hot data blocks can be identified by AMS easily**
  - **AMS can handle *deletions***
  - **Heavy hitters can be managed in a heap separately**



Hot Separation by AMS

- For low frequency items
  - Accuracy of approximation improves without heavy hitters





Static Wear Leveling

- Still, a big question remains for Static WL
  - Identify cold data blocks, not only the hot ones
  - Biased (or underestimating) approximations are useless
- AMS + BET?
  - When a triggering condition is satisfied, look for un-erased blocks from BET
  - Inquire the AMS Sketch for the approximate erase count of each un-erased block
  - Quickly find cold data blocks if the number of un-erased blocks is small; further evaluation is in order

Concluding remarks

- TB-scale flash storage devices are on the horizon
  - Scalable and economical designs are must
- Algorithmic innovations will make a difference
  - That's where real competitiveness comes from!