

SSD Trim Commands Considerably Improve Overprovisioning

Tasha Frankie

tcvanesi@ucsd.edu

Gordon Hughes Ken Kreutz-Delgado









- A model of trim is proposed and analyzed
- Such models are useful to reduce costs
- Using our model, we show improvement in the level of effective overprovisioning for uniformly distributed workloads





- Introduction
- Overprovisioning
- Trim Command
- Trim Performance Model
- Workload Model
- Theoretical Results
- Conclusion







- SSDs are increasingly important
 - Ubiquitous in embedded devices
- Erase-before-write requirement problems:
 - In-place writes are impractical, so dynamic logical-to-physical mapping is used
 - Write amplification due to garbage collection
- Trim command helps reduce penalties caused by erase-before-write







Overprovisioning/Spare Factor

- Overprovisioning: Putting more physical blocks on a device than user is allowed to access
 - Increases speed of device by reducing number of writes needed in garbage collection
 - Increases lifetime of device by spreading wear over more physical blocks
- Spare factor: $S_f = \frac{(T_p u)}{T_p}$ Range: (0, 1)
 - T_p = raw storage capacity of device, in pages
 - u = number of pages user is allowed to utilize

(Hu, 2009)







- Trim*: declares logical blocks inactive
 - Allows garbage collection to skip copying of trimmed physical pages when reclaiming space
 - Reduces the number of in use LBAs**
 - LBA is in-use when its most recent request was a write
 - LBA is not in-use if LBA has never been written, or if most recent request issued for it is a trim

* INCITS Working Draft T13/2015-D Rev.

★ LBA = Logical Block Address







Memory Teaser Results

- 25% of requests as trim transforms an SSD with zero specification spare factor (S_f) into one having a mean effective spare factor (S_{eff}) of 33%
 - This level of overprovisioning without a trim command would require 50% more physical pages than the user is allowed to write!





Memory Trim Performance Model

Assumptions:

- One LBA is same size as one physical page
 - Straightforward calculation of effective spare factor using number of in use LBAs at any time
- Only write and trim requests are considered
 - Read requests do not affect the number of inuse LBAs or the write speed of device







Memory Trim Performance Model

Markov Birth-Death Chain:

- State, X_n: number of in-use LBAs at time n
- Trim request: reduces state by 1, occurs with probability q_x
- Write request:
 - Leaves state unchanged (request is for an in-use LBA), occurs with probability r_x

or

 Increases state by 1 (request is for a not in-use LBA), occurs with probability p_x







Memory Formal Markov Model

Transition probabilities:

$$P\left(X_{n+1} = x - 1 \mid X_n = x\right) = q_x$$

$$P\left(X_{n+1} = x \mid X_n = x\right) = r_x$$

$$P\left(X_{n+1} = x + 1 \mid X_n = x\right) = p_x$$

- subject to $q_x + r_x + p_x = 1$

Unnormalized steady-state occupation:

$$\pi_x = \begin{cases} \frac{p_0 \cdots p_{x-1}}{q_1 \cdots q_x} & x \ge 1\\ 1 & x = 0 \end{cases}$$

(Hoel 1972)







sh Memory Workload Model

Uniform random workload

- Write requests uniformly random over all u user LBAs
- Trim requests uniformly random over all in-use LBAs
- Trim requests happen with probability q;
 Write requests happen with probability 1 q
- Unnormalized steady-state occupation

$$\pi_{x} = \left(\frac{1-q}{q}\right)^{x} \frac{u!}{u^{x} (u-x)!}$$







Steady-State Results for u >> 1

Gaussian distribution for number of in-use LBAs*

$$- \operatorname{Mean} = u\left(\frac{1-2q}{1-q}\right)$$

– Variance =
$$u\left(\frac{q}{1-q}\right)$$

* By an asymptotic expansion. Will happily share math details offline.





Steady-State Results (u = 1000)







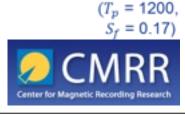
Santa Clara, CA



sh Memory Effective Spare Factor Seff

$$S_{eff} = \frac{T_p - X_n}{T_p}$$

- T_p = number of physical pages in device
- X_n = number of in-use LBAs at the current time n







Mean and Variance of Effective Spare Factor

- Mean Effective Spare Factor \overline{S}_{eff}
 - Can be expressed in terms of the specified spare factor S_f:

$$\overline{S}_{eff} = \left(\frac{1-2q}{1-q}\right) \left(\frac{q}{1-2q} + S_f\right)$$

- Variance
 - Depends on the size of the device in pages, T_p:

$$\operatorname{Var}(S_{eff}) = \frac{1}{T_p^2} \operatorname{Var}(X_n) = \frac{1}{T_p^2} u \left(\frac{q}{1 - q} \right)$$

Flash Memory Summit 2011 Santa Clara, CA





Memory Mean Effective Spare Factor

25% trim factor transforms an SSD with zero specified spare factor into one having an effective spare factor of 33%.

Without trim, this spare factor requires 50% more physical pages than the user is allowed to write!







- Trim performance models can allow manufacturers and customers to minimize amount of necessary physical overprovisioning
 - Save \$\$\$!







References

- 1. Hoel, P. G., Port, S. C., and Stone, C. J. (1972), Introduction to Stochastic Processes, New York: Houghton-Mifflin.
- 2. X. Y. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka, "Write amplification analysis in flash-based solid state drives," in Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, pp. 1-9, 2009.
- INCITS Working Draft T13/2015-D Rev. 7, "Information Technology - ATA/ATAPI Command Set - 2 (ACS-2)," June 22, 2011.











