Оценка зависимости износа и скорости записи твердотельного накопителя от размера резервной области и размера блока

Снегова Е.А., Алисейчик П.А., Алексеев Д.В.

В данной работе производится оценка зависимости избыточности записи (Write Amplification) от размера резервной области памяти (Overprovision) твердотельного накопителя (SSD) при разных значениях числа секторов в блоке N. Сборка мусора, т.е. перезапись частично заполненных блоков производится в соответствии с жадным алгоритмом (Greedy Garbage Collection).

Получена более точная формула чем в [1], поскольку отсутствует предположение $N \to \infty$. Показано, что при фиксированном размере резервной области избыточность записи является монотонно возрастающей функцией от N.

Ключевые слова: избыточность записи, резервная область памяти, сборщик мусора, твердотельный накопитель.

1. Введение

Физическое пространство памяти SSD-накопителя состоит из блоков, каждый из которых состоит из одинакового колисества страниц. Например, один блок типичного SSD-накопителя может содержать 512 страниц, при этом каждая страница может иметь объем 16 Kb. Единицей записи является страница, в то время как единицей стирания является многостраничный блок.

В силу ограничений технологии SSD, для того, чтобы записывать новые данные на страницу, страница должна быть очищена, т.е. предварительно требуется стереть весь блок, к которому она относится. Поэтому обновление данных, записанных во флеш-память происходит путем записи обновленных данных в новую область, причем область, в которую была записана старая версия подвергается процессу *инвалидации*,

т.е. страницы, содержащие старые данные помечаются как *невалидные*. Изначально все страницы считаются валидными. *Валидностью блока* называют количество содержащихся в нем валидных страниц.

При этом контроллер хранит и постоянно обновляет таблицу соответствия логических адресов физическим. Со временем количество невалидных страниц растет, а валидность блоков, соответственно, уменьшается. Контроллер периодически проводит стирание блоков, чтобы освободить место для записи. В процессе стирания производится копирование всех действительных (т.е. не подвергшихся инвалидации ранее) страниц в стираемом блоке.

Эффективность этого процесса стирания может существенно повлиять на производительность памяти, в частности, за счет дополнительного копирования может снизиться пропускная способность. Это явление называют *эффектом избыточной записи*. Избыточность записи определяется как среднее значение числа записей на SSD (в секторах), приходящееся на один сектор пользовательского запроса на запись. Избыточность записи влияет как на скорость записи, так и на износ физического носителя.

Для хранения повторно записанных страниц используется резервная область памяти — та часть физического пространства SSD, которая не доступна пользователю. Ее размер измеряется как отношение разницы размера физического пространства и логического пространства к логическому пространству (все составляющие этой формулы измеряются в секторах или байтах).

Задача оценки зависимости избыточности записи (Write Amplification, далее обозначается как W) от размера резервной области памяти (Overprovision, далее обозначается как R) твердотельного накопителя (SSD) при разных значениях числа секторов в блоке N возникла при программировании контроллера реального SSD-устройства. Предполагалось, что данный функционал должен работать с различными видами SSD-устройств, в частности, отличающимися числом секторов в блоке N. Таким образом, данная формула призвана проверить, что контроллер работает оптимальным образом, позволяя на случайном равномерно-распределенном в логическом пространстве потоке запросов получать минимально возможное значение избыточности записи.

В работе [6] была предложена вероятностная аналитическая модель для сборщика мусора со скользящим окном, был представлен метод вычисления избыточности записи. В [2] был получен результат, в котором зависимость избыточности записи от размера резервной области исходя

из предположения, что валидность блока равномерна распределена на отрезке $[v_{min}, N]$, где v_{min} — это минимальная валидность блока (блок с такой валидностью попадает в GC при жадном алгоритме "сборки мусора"), а N — число секторов в блоке. Полученная оценка избыточности записи не зависит ни от v_{min}, ни от N. В работах [4] и [8] независимо был получен аналогичный результат в предположении, что распределение валидности зависит от числа валидных страниц, что более точно соответствует действительности. Во всех этих работах используется сборщик мусора, использующий жадный алгорим. В работе [1] получена оценка $W(R)$ с использованием функции Ламберта. Эта оценка также не зависит от N, но является наиболее близкой к полученной в настоящей статье оценке $W(R, N)$, так как при при $N- > \infty$ $W(R, N)$ сходится к оценке $W(R)$ из [1].

2. Постановка задачи и результат

Далее опишем типичный упрощенный алгоритм работы флеш-контроллера. Память флеш контроллера состоит из трех основных компонент.

1) кэш объемом не менее одного блока;

2) таблица Block(lsa), в которой каждому логическому сектору (lsa) сопоставлен физический номер блока, в который он записан, или сопоставлено число -1, если данный сектор не используется;

3) таблица валидности блоков Validity(block), в котроой каждому блоку сопоставлена его валидность, то есть число таких секторов (lsa), у которых Block(lsa) = block.

Таблицы контроллера Block(lsa) и Validity(block) по сути составляют *систему трансляции адресов* (Flash Translation Layer, FTL).

Типичный алгоритм контроллера выглядит следующим образом. От управляющего устройства поступает запросна запись последовательного участка логических секторов. Постепенно эти сектора попадают в кэш. При этом «старые» данные записанные в данном логическом секторе становятся невалидными, поэтому значение Validity(Block(lsa)) уменьшается на единицу, а затем Block(lsa) присваивается значение -1.

Если в кэше оказалась хотя бы одна полная страница, то происходит запись данных из кэша на SSD. Для этого либо берется текущий открытый блок и записывается следующая подряд идущая страница, либо

сначала ищется пустой блок, объявляется текущим открытым блоком, а затем в него записывается первая страница.

Обозначим текущий открытый блок как *cur_block*. Тогда для каждого сектора lsa, поступившего из кэша на SSD, таблицы FTL менеяются следующим образом: $Block(lsa) = cur\_block$, $Validity(cur\_block) += 1$.

Поиск пустого блока — задача части контроллера под названием сборщик мусора (Garbage Collector). В настоящей статье рассматривается "жадный алгоритм"GC, при котором GC берет блок с наименьшей валидностью и все валидные данные помещает к кэш, а сам блок стирает, освобождая для новых записей.

Очевидно, что такой алгоритм работы контроллера возможет только в ситуации, когда объем памяти SSD (физическое пространство) превышает совокупный объем логического пространства.

Ниже приводится оценка избыточности записи W от объема резервной памяти R и числа секторов в блоке N. Верны следующие оценки:

1) Пусть $R_k = \frac{N}{N-k} * [\frac{1}{k+1} + ... + \frac{1}{N}] - 1$ $(k = 0, ...N-1)$, тогда

$$W(R_k) = \frac{N}{N-k}.$$

2) При $R \in [R_{k+1}, R_k]$ выполнено равенство

$$\frac{R - R_{k+1}}{R_k - R_{k+1}} = \frac{W - W_{k+1}}{W_k - W_{k+1}},$$

то есть W линейно зависит от R.

3) Если $R > \frac{1}{2} + .. + \frac{1}{N}$, то $W = 1$.

При разных значениях N зависимость $W(R)$ выглядит следующим образом (см. рис. 1-3):

Замечание: при при $N \to \infty$ приведенная выше оценка $W(R, N)$ сходится к оценке $W(R)$ из [1].

3. Оценка зависимости $W(R, N)$ в узловых точках

Обозначим логическое пространство за единицу, а физическое — за $1+R$. Без ограничения общности, будем считать, что размер сектора совпадает с размером страницы. Данное предположение упрощает выкладки, но никак не влияет на полученный результат. Кроме того, будем считать, что в одну единицу времени приходит одно из следующих двух событий:

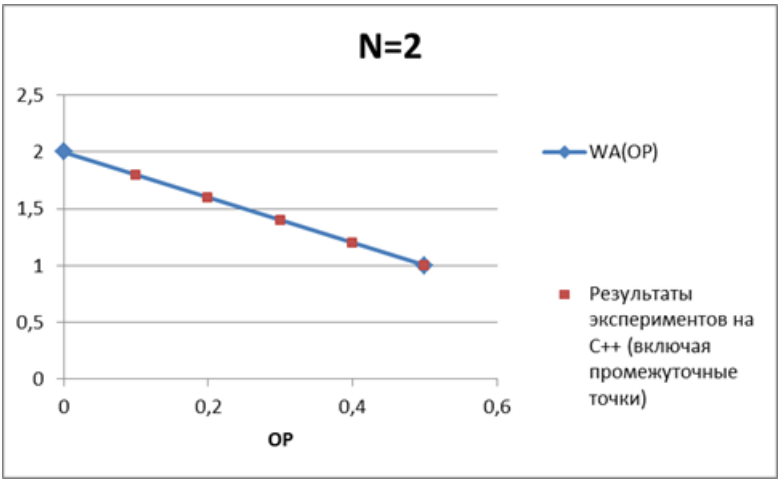

Рис. 1.

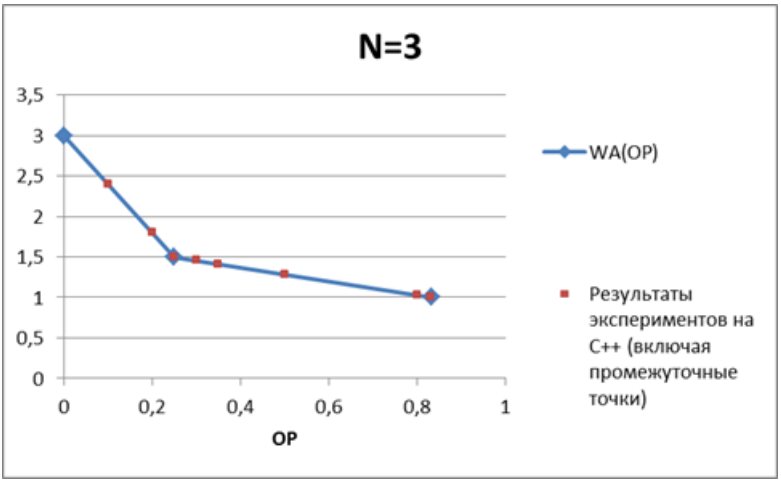

Рис. 2.

1) Выполнение одного запроса на запись.

   Поступает один запрос (то есть соответствующие данные попадают в кэш) и происходит инвалидация "старых" данных (изменение таблиц Validity и Block). Кроме того, если в кэше набралась одна страница секторов, то она записывается на SSD (в физическое пространство) в тот же самый момент времени.

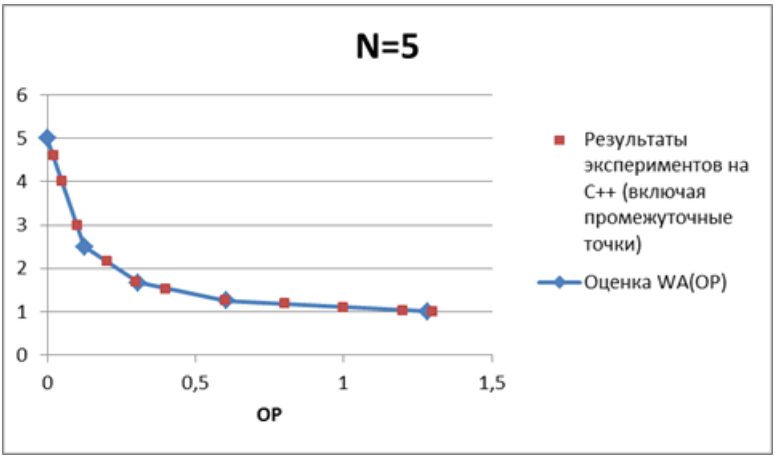

Рис. 3.

2) Работа GC — "превращение блока с минимальной валидностью в блок с валидностью N".

   Если в физическом пространстве не осталось больше свободного места, то есть записанная на предыдущем шаге страница была последней в открытом блоке, то GC ищет блок с минимальной валидностью, все валидные данные переписывает в кэш, блок очищает и объявляет его новым открытым блоком. В случае если минимальная валидность не нулевая, то в кэше оказывается минимум один сектор. Тогда все данные из кэша переписываются в физическое пространство, частично заполняя новый открытый блок.

Пусть k — минимальная валидность, C — скорость, с которой GC "превращает блок с минимальной валидностью в блок с валидностью N X_n — это доля блоков с валидностью n в общем числе блоков, составляющих физическое пространство, то есть верно, что

$$X_{k+1} + X_{k+2} + ... + X_N = \frac{1+R}{N}. \quad (1)$$

Используем предположение о равномерности распределения пользовательских запросов в логическом пространстве. Вероятность увеличения числа блоков с валидностью $n = k+1, .., N-1$ равна произведению числа блоков с валидностью $n+1$ - X_{n+1}, и размеру блоков с валидностью $n+1$, то есть самому числу $n+1$. Аналогично, вероятность уменьшения числа блоков размера n равна произведению числа блоков с валидностью

n - X_n и размеру блока n. Таким образом,

$$X'_n = X_{n+1}(n+1) - X_n \cdot n.$$

Уравнения, описывающие работу системы:

$$X'_k = X_{k+1}(k+1) - C$$

$$X'_{k+1} = X_{k+2}(k+2) - X_{k+1} \cdot (k+1) \tag{2}$$

$$.....$$

$$X'_N = C - X_N \cdot N.$$

В равновесной ситуации верно, что

$$X'_k = X'_{k+1} = ... = X'_N = 0. \tag{3}$$

Кроме того, $X_k = 0$, так как если $X_k > 0$, то с ненулевой вероятностью $k-1$ будет минимальной валидностью, что противоречит нашему изначальному предположению. Из (1), (2) и (3) следуют условия на R, при котором минимальная валидность блока равняется k:

$$R = \frac{N}{N-k} * [\frac{1}{k+1} + ... + \frac{1}{N}] - 1, \tag{4}$$

W в этом случае определяется следуюбщим образом:

$$W = \frac{N}{N-k}.$$

4. Оценка завистмости $W(R, N)$ в промежуточных точках

Осталось показать, что в промежуточных точках зависимость $W(R, N)$ линейно по R при фиксированном N.

Перепишем уравнения, описывающие работу системы для случая, когда минимальная валидность может равняться k или $k+1$.

$$\frac{\triangle X_k}{\triangle t} = X_{k+1}(k+1) - C * I_{X_k \neq \emptyset}$$

$$\frac{\triangle X_{k+1}}{\triangle t} = X_{k+2}(k+2) - X_{k+1} * (k+1) - C * I_{X_k = \emptyset}$$

$$\frac{\triangle X_{k+2}}{\triangle t} = X_{k+3}(k+3) - X_{k+2} * (k+2)$$

....

$$\frac{\triangle X_N}{\triangle t} = C - X_N * N$$

Пусть в α — доля случаев, когда минимальная валидность блока k, а в $1-\alpha$ случаях - минимальная валидность $k+1$, $\alpha \in [0,1]$.

Тогда

$$C = \frac{1}{N-k-(1-\alpha)},$$

$$W = \frac{N}{N-k-(1-\alpha)}.$$

$$X_{k+1} + X_{k+2} + ... + X_N = \frac{1+R}{N} \Rightarrow$$

$$\frac{1}{N-k-(1-\alpha)} * [\frac{\alpha}{k+1} + \frac{1}{k+2} ... + \frac{1}{N}] = \frac{1+R}{N}, \tag{5}$$

откуда следует, что

$$\alpha = \frac{[\frac{1}{k+2} + ... + \frac{1}{N}] * N - (N-k-1) * (1+R)}{1+R-\frac{N}{k+1}},$$

$$W = \frac{N * [1+R-\frac{N}{k+1}]}{(N-k-1) * [-\frac{N}{k+1}] + [\frac{1}{k+2} + ... + \frac{1}{N}] * N},$$

то есть уравнение линейно по R.

5. Заключение

Полученные оценки были проверены путем симуляции работы твердотельного накопителя. Показано, что в случае конечного числа секторов они являются более точными, чем оценка из [2].

В дальнейшем планируется обобщение полученного результата на случай неравномерного распределения частоты обновления данных. Т.е. когда данные делятся на "горячие" (т.е. часто обновляемые) и "холодные" (редко обновляемые).

Список литературы

[1] Stoica R., Ailamaki A. Improving flash write performance by using update frequency //Proceedings of the VLDB Endowment. – 2013. – Т. 6. – №. 9. – С. 733-744.

[2] Agarwal R., Marrow M. A closed-form expression for write amplification in NAND flash //GLOBECOM WorkshRs (GC Wkshps), 2010 IEEE. – IEEE, 2010. – С. 1846-1850.

[3] Sah C. T. Fundamentals of solid state electronics. – World Scientific Publishing Company, 1991.

[4] Desnoyers P., Analytic modeling of SSD write performance // SYSTOR '12 Proceedings of the 5th Annual International Systems and Storage Conference, Article No. 12

[5] Desnoyers P., Analytic Models of SSD Write Performance // ACM Transactions on Storage, 2012.

[6] Hu X.-H., Eleftheriou E. , Haas R., Iliadis I., Pletka R. Write Amplification Analysis in Flash-Based Solid State Drives // SYSTOR, 2009.

[7] Yang Y., Zhu J., Analytical modeling of garbage collection algorithms in hotness-aware flash-based solid state drives // Mass Storage Systems and Technologies (MSST), 2014, 30th Symposium on.

[8] Xiang L., Kurkoski B.M., An improved analytic expression for write amplification in NAND flash // Computing, Networking and Communications (ICNC), 2012 International Conference on

[9] Wang W.-N., A simplified Model of Write Amplification for Solid State Drives Adopting Page level Address Translation Mechanism // ICEEAC, pp.2156-2160, 2010.

The estimation of dependency SSD wear level and write speed on overprovision and block size
Snegova E.A., Aliseychik P.A., Alexeev D.V.

This paper is devoted to estimation of SSD write amplification dependency on R — overprovision and N — number of sectors in a block. Garbage collector uses greedy algorithm.

More precise formula than in[1] is obtained because it does not depend on $N \to \infty$ assumption. Proved that for fixed overprovisioning the write amplification is monotonic increasing function of N.

Keywords: Write amplivication, overprovision, SSD, garbage collector.