BCW on HDDs for Hybrid Storage Server

SRI 2021 Team 1

Nikhil Mehta(201801030)
Vatsal Gujarati(201801162)

Introduction

1) Hybrid Storage servers are used for their cost effectiveness and µs level responsiveness.
2) HDDs are underutilized and SSDs are overutilized. This leads to a fast wear out in SSDs due to frequent garbage collections induced by the intensive writes in SSDs.
3) HDDs exhibit a periodic staircase shaped pattern of write latency resulting from the buffered writes in HDDs. This suggests that HDDs can as well provide µs level write delay which is close to SSDs write performance.
4) So the goal is to exploit the performance of HDDs to absorb an many writes as possible to avoid SSD overuse without performance degradation.

a) Storage clouds have deployed hybrid storage because of the high performance of SSDs and large capacity of HDDs.
b) Generally HDDs are underutilized and writes are unfriendly to SSDs because
    1) SSDs have limited P/E (Program/Erase) cycles [6, 36] that are directly related to the amount of writes.
    2) SSDs suffer from unpredictable, severe performance degradations resulting from garbage collections.

c) Therefore, a series of continuous and sequential small HDD writes (e.g., 4KB) exhibit low latency (e.g., 35μs) for about 60 ms, and then a sharply elevated high latency (e.g., 12ms), which is followed by medium latency (e.g., 55μs) for about 40ms. The three states of write behaviors, or write states in short, are referred as **fast, mid, and slow** writes, respectively. The former two types of writes can **provide μs-level responsiveness** because the buffer is not full.(1)

d) The key challenge for adopting buffered writes in HDDs to take advantage of the fast and mid writes is the difficulty in predicting precisely when these write states would occur.(2)

e) So we built a prediction model for sequential and continuous write patterns to predict future states. The prediction of next write state can be achieved with the information of buffered-write period and current write state.

f) Then, we use a Buffer-Controlled Write (BCW) approach. BCW can proactively and effectively control the buffer write behavior according to the predictor and runtime IO monitoring. Besides, BCW also actively “skip” slow writes by filling padded data during HDD slow writes.

g) Thereafter, we use a mixed IO scheduler (MIOS) for SSD-HDD hybrid storage by leveraging the BCW approach. MIOS adaptively redirects incoming writes to SSDs or HDDs depending on write states, disk status.

SSD-HDD Hybrid Storage

a) To accommodate exponentially increasing storage requirement while achieving overall cost-effectiveness, SSD-HDD hybrid storage has emerged to be an inevitable choice for cloud providers.
b) Most cloud providers expect larger storage capacity and better performance but at lower cost. To meet this demand, they increasingly embrace storage heterogeneity, by deploying SSDs, which offer lower IO latency, as the primary tier and HDDs, which provide larger capacity at low cost as the secondary tier.
c) The fast SSD tier generally plays the role of a write buffer to quickly persist incoming write data, which are eventually flushed to the slower but larger HDD tier.

Write Behavior of Hybrid Storage

a) Write-intensive workloads widely exist in many production environments, such as enterprise applications, supercomputing, and clouds.

b) It was observed that on Pangu (distributed large-scale storage platform and provides cost-effective and unified storage services for Alibaba Cloud), the workload analysis was as follows.

   1) Most requests are writes. The amount of data written is 1-2 orders of magnitude larger than that of data read from them. Actually, nearly 3 TB data are written to every SSD each day, which is close to DWPD (Drive Writes Per Day) that strictly limits the amount of SSD data written daily for reliability.
   2) The amount of data written to SSDs and HDDs differ dramatically. For instance, the average SSD IO utilization is up to 28.5% while it is less than 10% in HDD. Even so, most of the HDD utilization is used in dumping SSD data, rarely servicing user requests.

Challenges

1) To relieve the SSD pressure from write-dominate workloads, a simple solution is to increase the number of SSDs in the hybrid nodes. However, this is a costly solution as it increases the total cost of ownership.
2) An alternative is to exploit the severely underutilized HDD capacity in hybrid storage nodes when SSDs are overused.
3) The state of the art solution **State-Write-Redirect(SWR)** redirects large SSD writes to idle HDDs.
4) However, the IO delays experienced by requests redirected to HDDs are shown to be 3-12 times higher than those experienced on SSDs. This is clearly undesirable, if not unacceptable, for most small writes that demand μs-level latency.
5) So the key challenge is how to reduce HDD IO delay to the μs-level that is close to SSDs.

HDD Write Behaviors

1) We'll perform continuous and sequential writes, which is the most friendly write pattern for HDDs so as to assess the possibility of achieving μs-level write latency on HDDs.

Buffered Writes

"Profiling" process is done to observe detailed HDD behaviors, a series of continuous and sequential writes with the same IO size are written to an HDD. We select five representative HDD products: West Digital 10TB, 8TB, 4TB, Seagate 8TB, 4TB. The 4TB Seagate HDD is SMR (Shingled Magnetic Recording) and the other four HDDs are PMR (Perpendicular Magnetic Recording).

a) For each tested HDD, the series of continuous sequential write requests experience a similar sequence of three level write latency, i.e., low, mid, and high latencies, forming a staircase-shaped time series.

b) HDD write behavior is periodic. At the beginning low latency writes last for a period, followed by a spike (high-latency writes) and then mid-latency writes. For continuous write patterns, high-latency writes and mid-latency writes appear alternately.

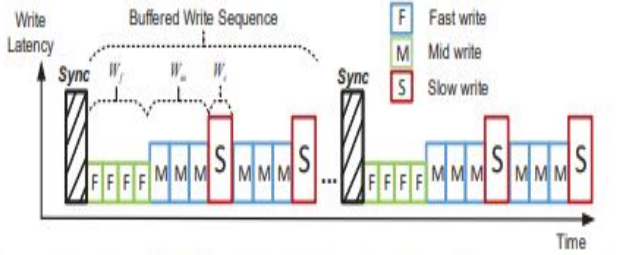


Figure 4: The HDD Buffered-Write Model with two complete Buffered Write Sequences.

Reason behind the observed HDD write behavior is because

a) Modern HDDs deploy a built-in DRAM (e.g., 256MB for the 10TB and 8TB HDDs).
b) But only part of the DRAM can be used as a buffer for the incoming IO, rest is used as a read cache.
c) The exact mechanism by which this built-in DRAM in HDD is used, which varies with the HDD model, is generally proprietary to the HDD manufactures only.

Upon successful buffering of a write, HDD immediately informs the host of request completion.

When the buffered data exceed a threshold, the HDD must force a flushing of the buffered data into their locations in the disk media.(3)

To explicitly empty the buffer, we can actively invoke sync() to force flushing.

HDD Buffered Write Model

1) A Buffered-Write Sequence consists of three types of HDD buffered writes, i.e., Fast (low-latency), Mid (mid-latency) and Slow (high-latency) writes, which denotes as F, M, and S, respectively.
2) The IO delays corresponding to the F,M,and S states are Lf, Lm, and Ls respectively as shown in the table.

F-state: an incoming write request *wi* with the size of si can be buffered completely in the built-in DRAM buffer of HDD.

M-state: This state means that the write buffer is close to be full.

S-state: This state means that the write buffer is full and any incoming write request is blocked.

Parameters	Description
$L_{f/m/s}$	The IO delays of write requests in the F/M/S write states
$W_{f/m/s}$	The cumulative amount of written data for the Fast/Mid/Slow Stages
$T_{f/m/s}$	The time duration of the Fast/Mid/Slow Stages
s_i	The IO size of write request i

3) A Buffered Write Sequence lasts a Fast stage, followed by one or more Slow-and-Mid stage-pairs. The sequence begins when there is sufficient buffer available for Fast stage (e.g., close to empty). It ends when current series of continuous writes ends.

4) The Fast, Mid, and Slow Stage last for Tf , Tm, and Ts respectively, which are determined by the cumulative amount of written data Wf , Wm, and Ws in the respective states. Actually, Wf = Tf ∗ si/Lf and it is applied to Wm.

Ex: In the 10TB HDD with 64KB write requests

Lf=180μs, Lm=280μs, Ls=12ms,Tf=60 ms,Tm=37ms,Ts=12ms.

Wf=16MB,Wm=8MB. Ws depends on the IO size si.

According to the HDD buffered-write model, the Fast and Mid writes of HDD have 100-μs-level latency, which can approach the write latency of SSDs.

5) This motivates us to design a controllable buffer write strategy for HDDs to reduce writes on SSDs in hybrid storage systems without sacrificing the overall performance

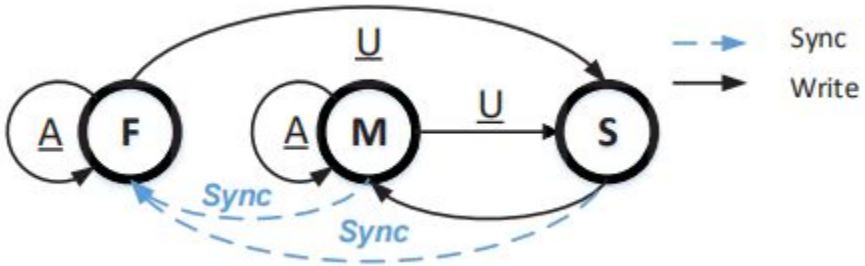
Design

To fully exploit HDD buffered writes, two critical challenges must be addressed:-

1) The first is how to determine which write state that a write request will be Fast (F), Mid (M), or Slow (S), in order to properly schedule the write request.

2) The second is how to steer an incoming write request to HDD without performance degradation.

Write - State Predictor



- F/A : The current write state is F and the buffer is available. Next write state is most likely to be F.
- F/U : Although the current write state is F, the buffer is unavailable. Next write state is likely to change to S.
- M/A : The current write state is M and the buffer is available. Next write state is most likely to remain M.
- M/U : Although the current write state is M, the buffer is unavailable. Next write state should be S.
- S : The current write state is S. Next write state will be M with a high probability.
- The *Sync* operation will force the next write state and buffer state back to be F/A in all cases.

Algorithm of Write - State Predictor

Input: Current write request size: *size*; The last write state: *state*; Current accumulative amount of data written: *ADW*;
The amounts of data written in the F state and M state: W_F and W_M
Output: Write-state prediction for the next request (F, M or S)

```
1:  function Predictor()
2:  if state == F then
3:      if (ADW + size) < W_f then : return F
4:      else: return S
5:      end if
6:  else if state == M then
7:      if (ADW + size) < W_m then : return M
8:      else: return S
9:      end if
10: else: return M
11: end if
```

Buffer - Controlled Write

Input: The max loop of Buffered Write Sequence: $Loop_{max}$
Request R_i size: $size_i$; Current written amount: ADW;
The state of last write: $state$;
Active padded writes and their size: PS,PF and $size_{PS}$,$size_{PF}$

```
1:  sync()
2:  while loop < Loop_max do
3:      if request R_i in the HDD write queue then
4:          write R_i to HDD, update ADW and state
5:      else
6:          if Predictor() == S then
7:              flag_HDD = False     // Stop receiving
8:              while state == S do
9:                  write PS to HDD, update ADW and state
10:             end while
11:             flag_HDD = True      // Start receiving
12:             reset ADW; loop++
13:         end if
14:         if Predictor() == M then
15:             write PF to HDD; update ADW and state
16:         end if
17:     end if
18: end while
```

Mixed IO scheduler

Input: SSD queue length at time t: $l(t)$;
Queue length threshold: L; HDD available flag: $flag_{HDD}$;
Schedule Strategy: $MIOS\_D$ or $MIOS\_E$

```
1: if (flag_HDD == True) then
2:     if l(t) > L && Predictor() == F or M then
3:         Send to HDD queue
4:     else if MIOS_E && Predictor() == F then
5:         Send to HDD queue
6:     else: Send to SSD queue
7:     end if
8: else: Send to SSD queue
9: end if
```

What have we done in SRI

- We first did a brief research on the topic of buffered controlled writes on HDD and on the topic of how to have a hybrid storage of SSD and HDD.

- After the study we wrote a code which could predict the state in which the HDD is currently in and the code efficiently manages the write operation on the SSD and HDD concurrently.

Learnings

- Our major learning included the understanding of the working of HDDs and SSDs. It helped us give a clear picture about the internal working and data management in SSDs.

- On top of that, we also learned why SSDs are preferred over HDDs.

- Major learning was that how the HDDs are underutilized and how they can be put in proper use to exploit them to their full potential thereby reducing the fast wear out of SSDs.

References

1) Wang, Shucheng, et al. "{BCW}: Buffer-Controlled Writes to HDDs for SSD-HDD Hybrid Storage Server." 18th {USENIX} Conference on File and Storage Technologies ({FAST} 20). 2020.

2) Gao, C., Shi, L., Zhao, M., Xue, C. J., Wu, K., & Sha, E. H. M. (2014, June). Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives. In *2014 30th Symposium on Mass Storage Systems and Technologies (MSST)* (pp. 1-11). IEEE.

3) Deng, F., Cao, Q., Wang, S., Liu, S., Yao, J., Dong, Y., & Yang, P. (2020, August). SeRW: Adaptively Separating Read and Write upon SSDs of Hybrid Storage Server in Clouds. In *49th International Conference on Parallel Processing-ICPP* (pp. 1-11).

4) Gokul Soundararajan, Vijayan Prabhakaran, Mahesh Balakrishnan, and Ted Wobber. Extending ssd lifetimes with disk-based write caches. In FAST, volume 10, pages 101–114, 2010.

5) Jiang, C., Han, G., Lin, J., Jia, G., Shi, W., & Wan, J. (2019). Characteristics of co-allocated online services and batch jobs in internet data centers: a case study from Alibaba cloud. *IEEE Access*, *7*, 22495-22508.

6) Pan Yang, Ni Xue, Yuqi Zhang, Yangxu Zhou, Li Sun, Wenwen Chen, Zhonggang Chen, Wei Xia, Junke Li, and Kihyoun Kwon. Reducing garbage collection overhead in {SSD} based on workload prediction. In 11th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 19), 2019.