STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers
Virtual Conference
September 28-29, 2021
A SNIA Event

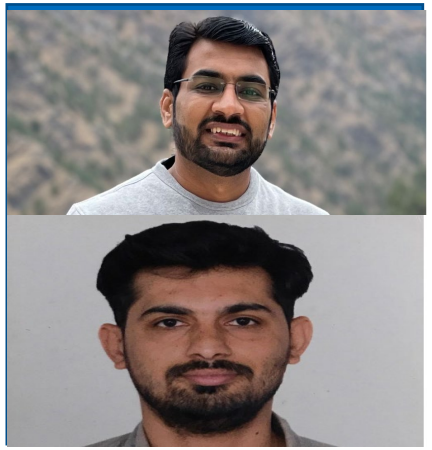
Unearthing the Impact of Sanitize on Performance and Latency in embedded storage

Sumeet Paul, Senior Manager
Shankar Athanikar, Senior Staff Engineer
Samsung Semiconductor India R&D

Disclaimer

- This presentation and/or accompanying oral statements by Samsung representatives collectively, the “Presentation” is intended to provide information concerning the embedded storage and memory industry and Samsung Electronics Co., Ltd. and certain affiliates (collectively, “Samsung”).While Samsung strives to provide information that is accurate and up-to-date, this Presentation may nonetheless contain inaccuracies or omissions. As a consequence, Samsung does not in any way guarantee the accuracy or completeness of the information provided in this Presentation.
- This Presentation may include forward-looking statements, including, but not limited to, statements about any matter that is not a historical fact; statements regarding Samsung’s intentions, beliefs or current expectations concerning, among other things, market prospects, technological developments, growth, strategies, and the industry in which Samsung operates; and statements regarding products or features that are still in development. By their nature, forward-looking statements involve risks and uncertainties, because they relate to events and depend on circumstances that may or may not occur in the future. Samsung cautions you that forward looking statements are not guarantees of future performance and that the actual developments of Samsung, the market, or industry in which Samsung operates may differ materially from those made or suggested by the forward-looking statements in this Presentation. In addition, even if such forward-looking statements are shown to be accurate, those developments may not be indicative of developments in future periods.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, do not assume any responsibility or liability for damages arising out of any reliance on or use of this information
- The type of Solutions mentioned here are the generic technologies available in market and test architecture discussed during this presentation are for illustration purpose only.

Outline:



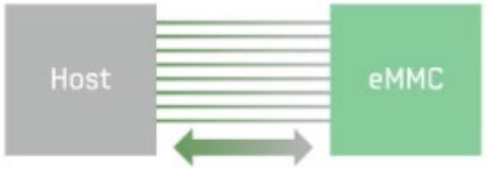
- Evolution of Embedded Storage.
  - Introduction and history
  - UFS is SSD for embedded hosts
- Garbage Collection and Its influence on Performance.
- Sanitize
  - Myths about Sanitize / Purge.
  - Sanitize for achieving QoS .
  - Experimental Data.
- Use of Discard as Mount Option .
- Conclusion.

Evolution of Embedded Storage :

Introduction and History

Evolution of Embedded Storage :

Parallel(x8) Interface



eMMC 5.1

400MB/s
Half Duplex communication.
Widely used standard before UFS2.1 adaptation.

UFS 2.1

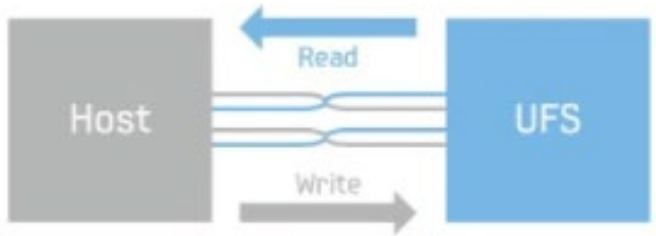
Gear 3 Lane 2.
Can achieve speed up to 1200MB/s over full duplex links
Industry started to adapt in flagship phones.

UFS 3.1

Gear 4 Lane 2.
Performance offering up to 2400MB/s double of UFS2.1
Extended temperature Support for Automotive

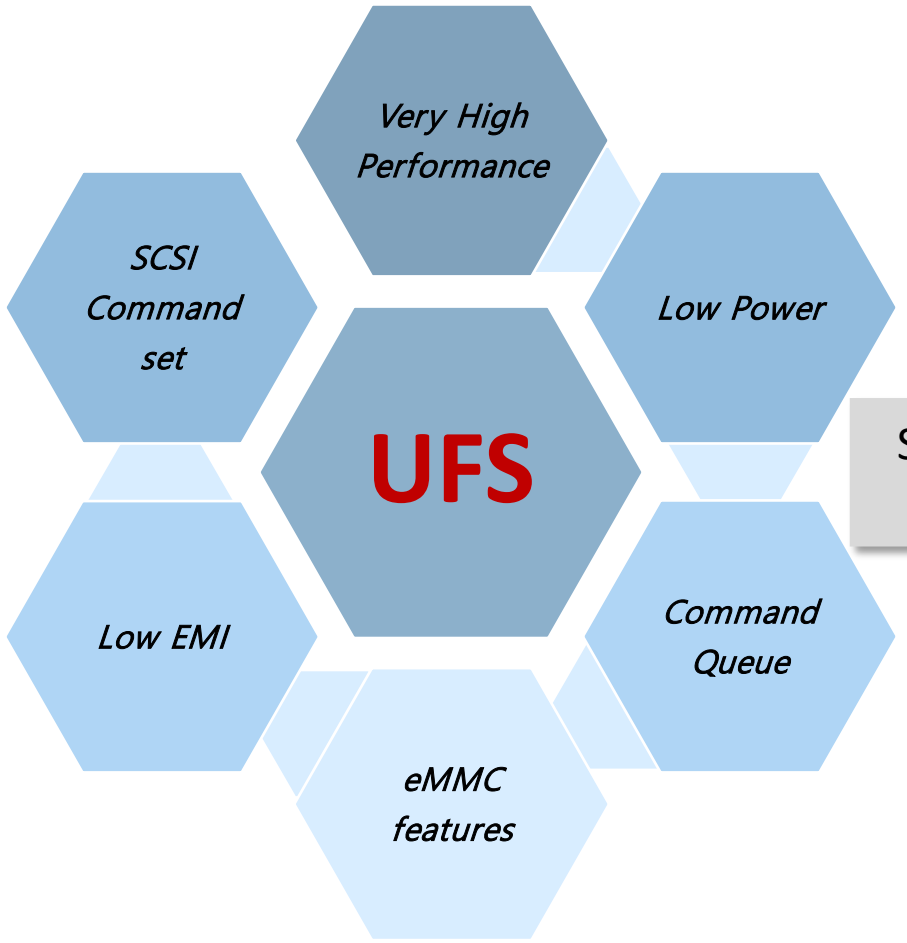
UFS 4.0

Gear 5 Lane 2.
Under Development
Future Products

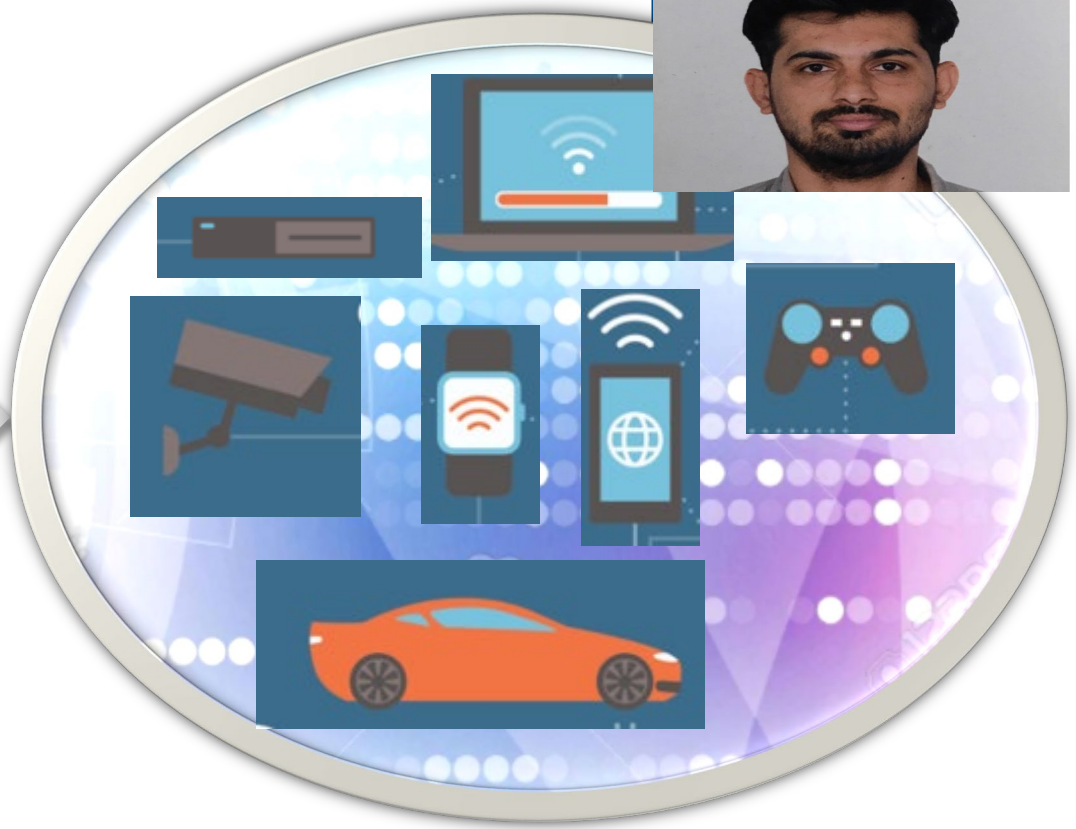


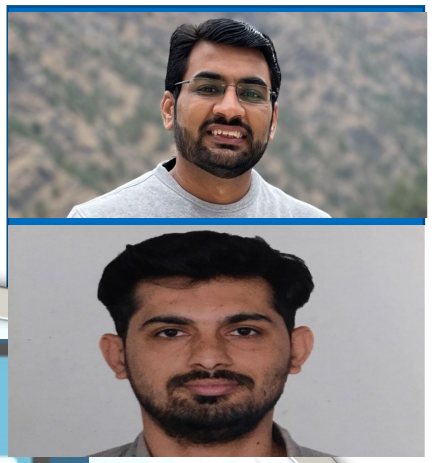
Full Duplex (Read & Write)

UFS is SSD for Multiple segments



SCSI + eMMC Features

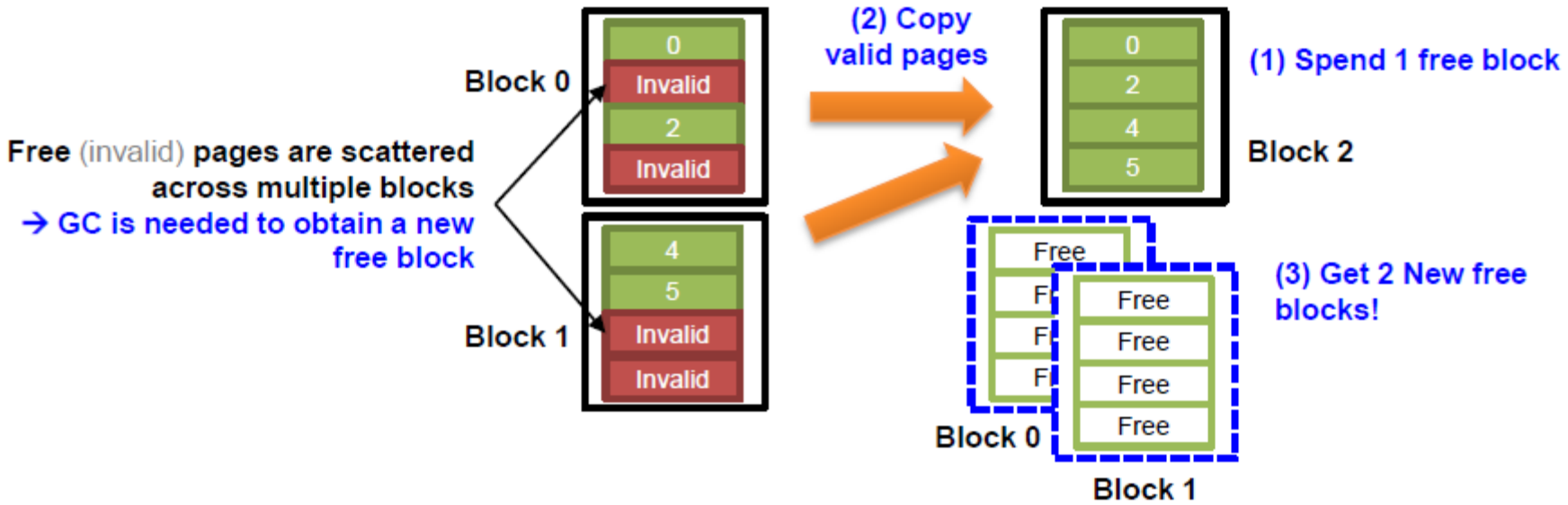




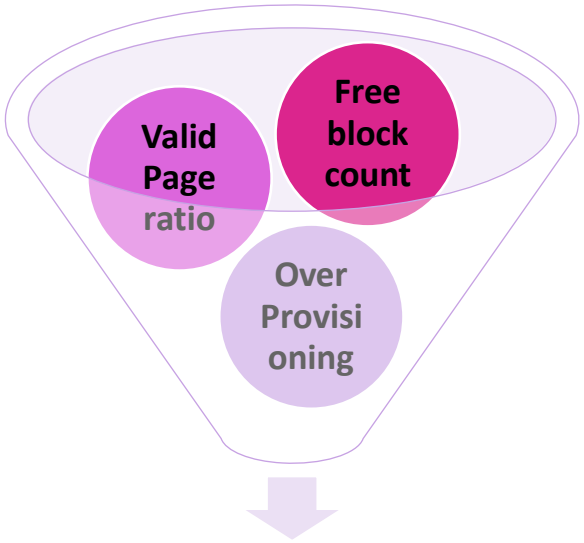
Garbage Collection and Its influence on Performance

Garbage Collection in nutshell

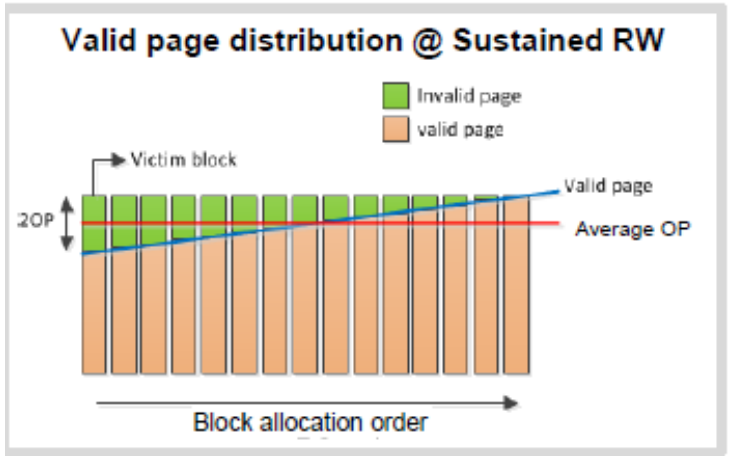
- What is Garbage? – Garbage is written data on Flash, which is no longer accessed by host
  - What makes Garbage? – Re-write or discard by host
- Garbage Collection (GC) is executed to obtain free blocks to write new data
  - Copy valid pages from Source Block(s) to Destination Block(s) to obtain a new Free Block
  - Source (= Victim)Block: A block with the lowest number of valid pages (= least amount of data to copy)



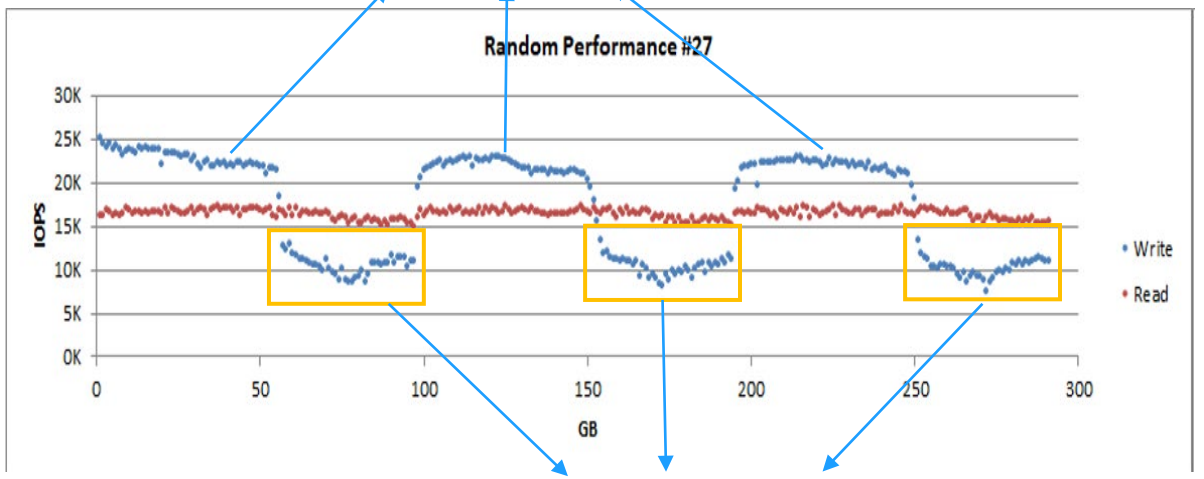
Garbage Collection: Major elements and their influence on Performance



Garbage Collection



Device not in GC Cond FB > Threshold



Device in GC Condition because of which there is performance drop FB < Threshold

OP ↑ →Valid Page % ↓→WAF ↓ →Perf ↑

Due to GC operation systems in real face increased application launch time, ANR, Lag in operation etc.

Sanitize

Myths about Sanitize / Purge.

Features of Sanitize

Use of Sanitize/purge only as Security Feature

In embedded storage (eMMC and UFS) commonly known purpose of sanitize/purge is to rendering the data on the media infeasible to recover for a given level of effort.

Sanitize(eMMC) and Purge(UFS) are host driven method to sanitize the device as per privacy requirements developed to meet standard guidelines by some governing bodies.

The use of the Sanitize operation requires the device to physically remove data from the unmapped user address space i.e. based on secure removal type

As part of sanitize ,FTL performs scan of each block and moves Valid pages to another Free block . Operation repeated until there is no invalid pages in user block and erase all unmapped blocks.

Sanitize and its concealed features

- Seasonable use of sanitize helps in reducing the latency, enhancing user experience with applications and improving the performance.
- Sanitize utilized in accordance with storage device policy will significantly improve QoS(Quality of Service) i.e. better consistency and predictability of latency (storage response time) and performance while serving read/write commands

Performance|Latency

Betterment in various mobile real world scenario i.e. App launch time, Better User eXperience etc.

Security

Choosing a right sanitization methodology so not only meta data get sanitize even whole user data is sanitize on disposal of host requirement of data security.

Lifetime

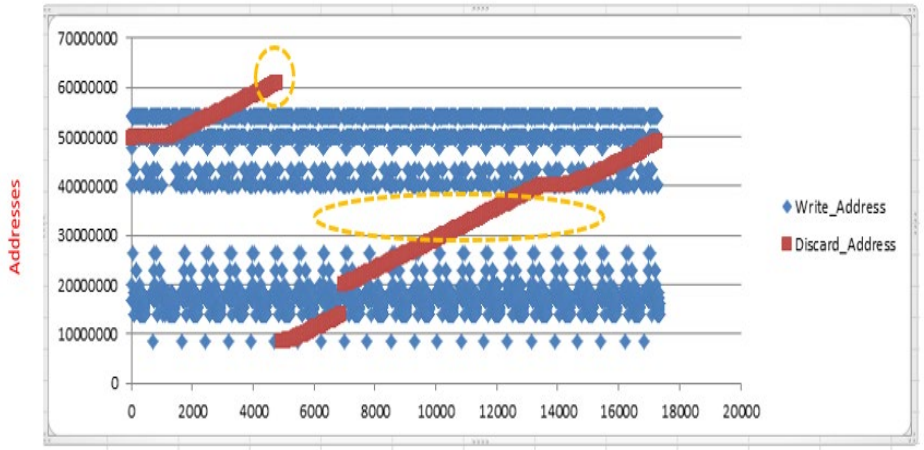
Since the region being operated on is not accessible by the host, applications requiring this feature must work with individual device manufacturers to ensure this operation is performing properly and to understand the impact on device reliability

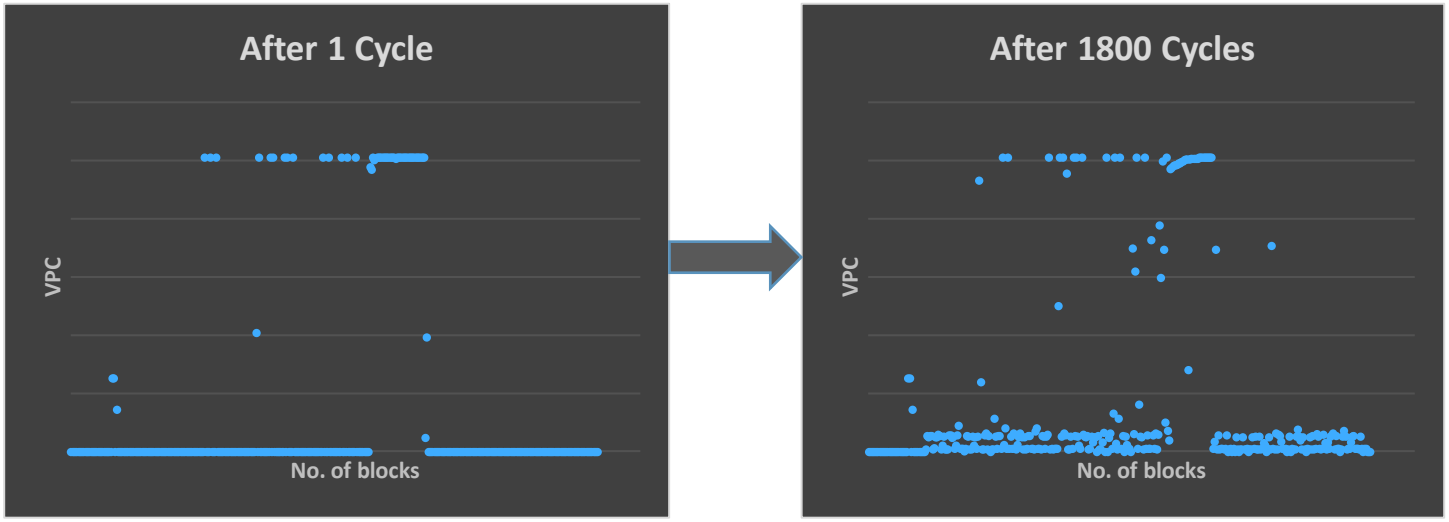
Real World Data:

Typical Automotive and Mobile Pattern

Automotive Pattern: Analysis

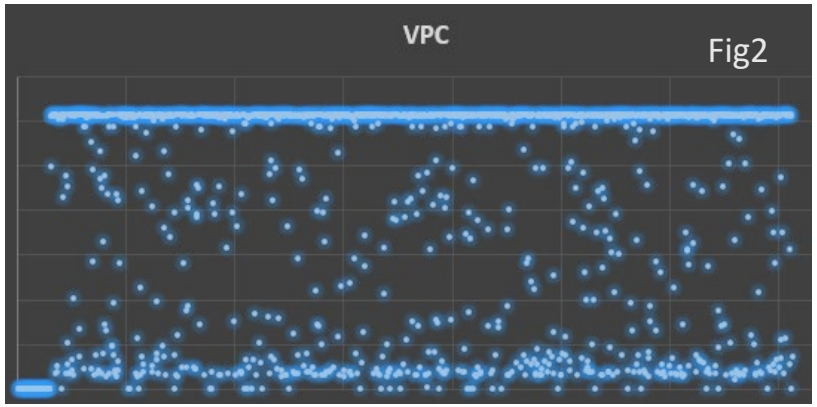
- Experiments were performed on state-of-the-art automotive chipset from Tier 1 customers.
- Here as precondition, host has written OS image to device and few of that area is used as read only. In some area, host keeps on updating the same address range while booting.
- First Image shows the written pattern of host, where after 50 cycle of boot there is discard performed on full user area. These overwrites creates lot of invalid space in underlying storage and later blind discard sent to full user area.
- Second image shows how continuous over write on certain range, creates fragmentation in device and varies the valid page ratio in each firmware block. Due this scattering of valid pages all over blocks, free block count becomes very less to cater upcoming writes and device has to perform urgent garbage collection





Mobile Pattern: Analysis

- Analysis is done on the Logs captured [I/O commands (Protocol Analyzer) and Meta data from the storage device] on the android mobile system .
- Host has pre-written OS image and that are is kind of RO. In some free area its is using for multi media data usage where user continuous writes are commencing.
- Fig1 indicates the various chunk size used by the host system .
  - Read chunks : ranging from : 4-1024KB
  - Write chunks : ranging From 4-1024KB
- As random chunk sizes are used, which has made valid pages(in Fig2) to distribute across the device causing FB(Fig3) count reduction in the device .
- As FB count is reduced below threshold forcing device to enter into GC condition to cater the next incoming writes effectively.



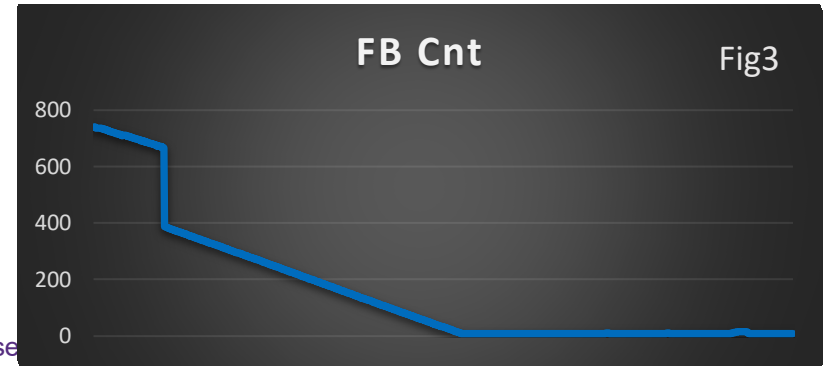


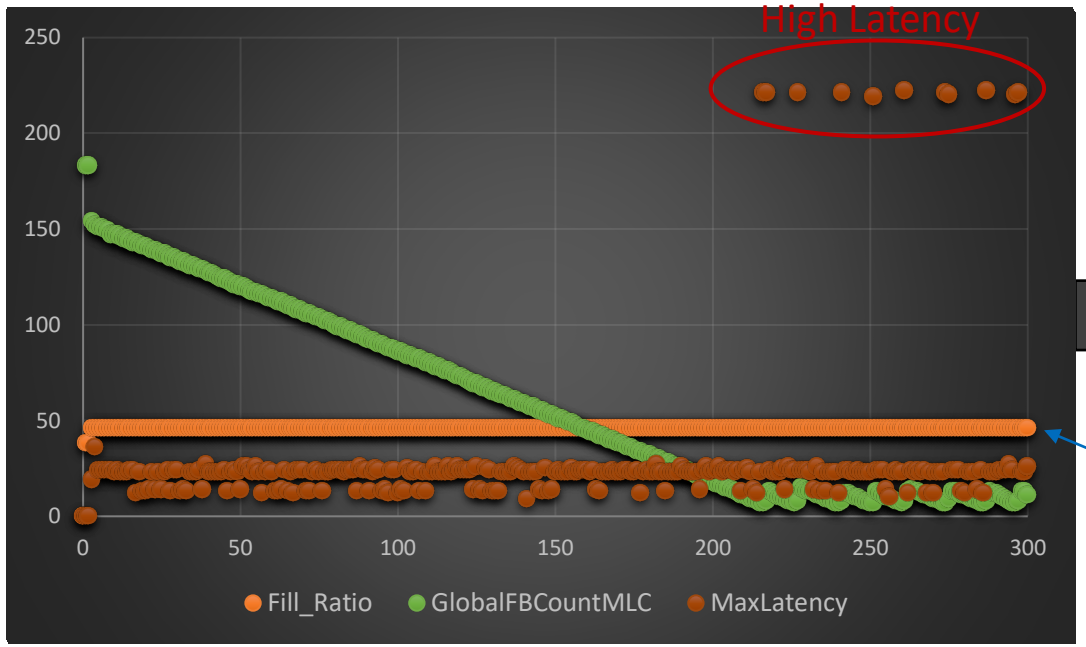
Fig3

Sanitize

Sanitize for Achieving QoS.

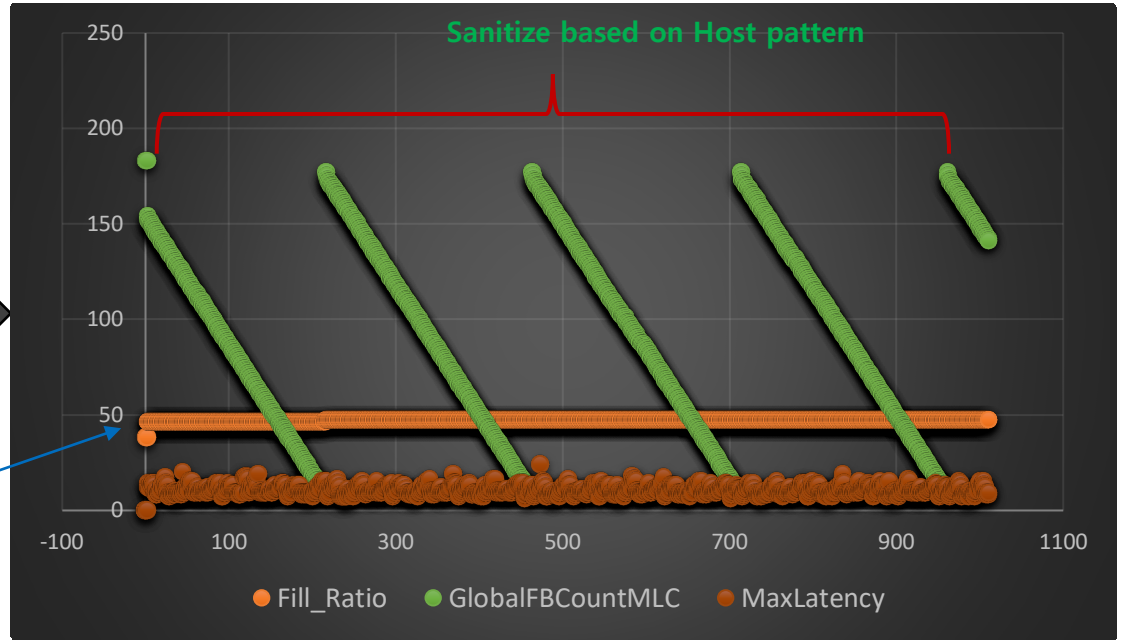
Sanitize: Based on host pattern and fill ratio

Here, shows how seasonable use of sanitize based on **host pattern and fill ratio** of device has made device to never get into garbage collection and host never faced any high latency and lag in user application



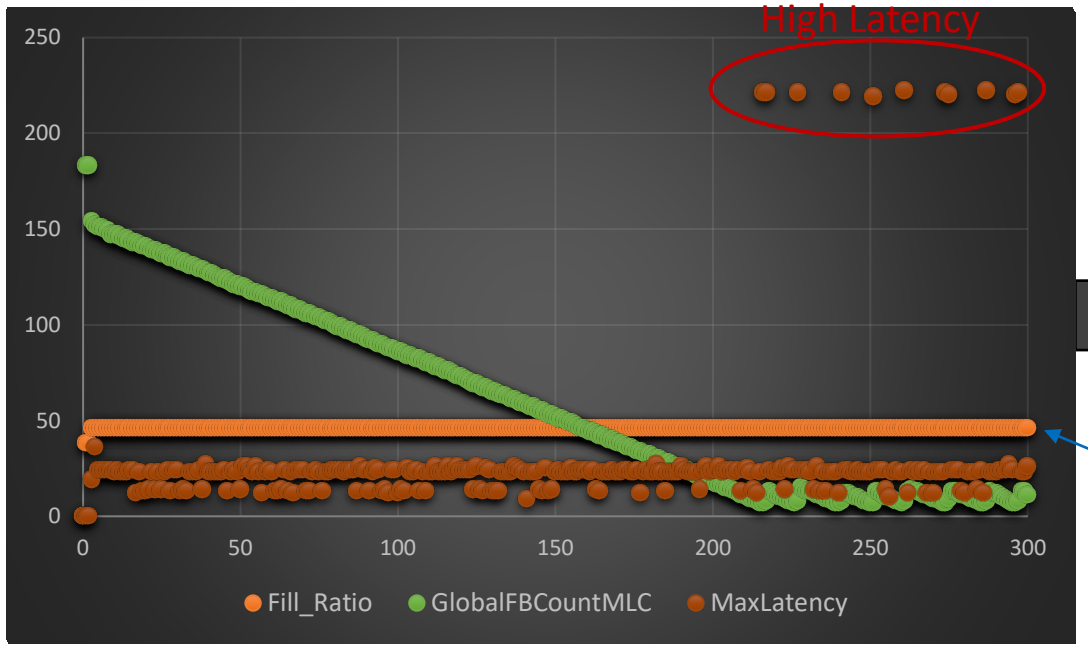
After Sanitize

Consistent fill ratio 50%



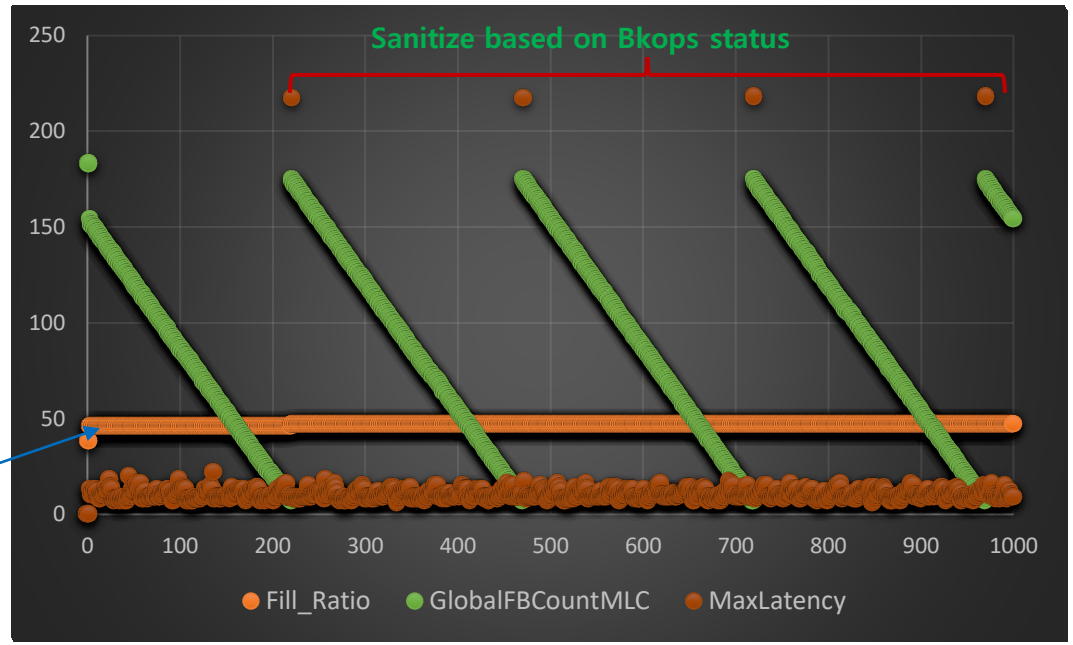
Sanitize: Based on BKOPS status flag

Here, how use of sanitize based on urgent **BKOPS status** of device has overcome this latency issue significantly. Here sanitize is issued by host ,once device raised urgent BKOPS flags and host has performed sanitize in idle time and subsequent write has not observed long latency.



After Sanitize

Consistent fill ratio 50%

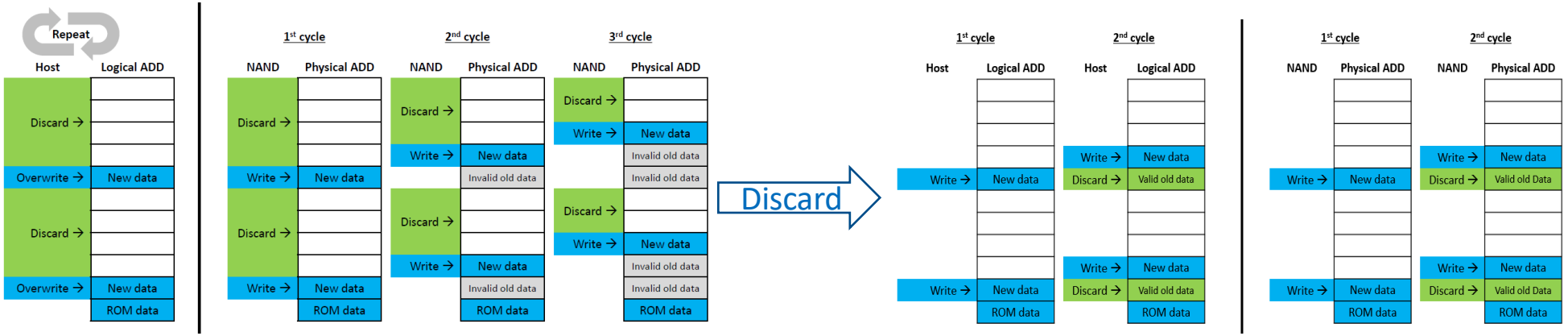


Discard as Mount Option

Use and Benefits

Use of Discard as Mount option

- If any file is deleted, Discard will be issued to underlying storage device.
- When user deletes file through Operating system, it is not physically deleted from the storage medium, as a matter of fact this file data is marked as Invalid but remains in the unmapped address space.



- Approaches to send the discard to mounted file systems.
  - fstrim .
  - “discard” as one of the mount parameters.
- Gains on storage side :
  - Increased Random write Performance .
  - Reduced Garbage Collection Cost .
  - Increase Life Time

Conclusion

Key Take Away

- How the close collaboration between host and storage device can improve the performance/latency and lifetime of embedded storage device is demonstrated.
- As a result, two main characteristics of underlying storage device is exhibited.
  - First, the use of Discard as mount option reduces the fragmentation significantly and eventually the BKOPS time and occurrence of garbage collection.
  - Second, seasonable use of Sanitize to improve the random write latency. As a result, we confirmed that our methods improves the random write latency of storage significantly.
  - As it is already established and being used by many customer and certain that the analysis and the proposed method can be used by an host developer to optimize the random write performance/latency and lifetime of storage device.

Thank You!

Please take a moment to rate this session.

Your feedback is important to us.

