SMR, the ZBC/ZAC Standards and the New libzbc Open Source Project

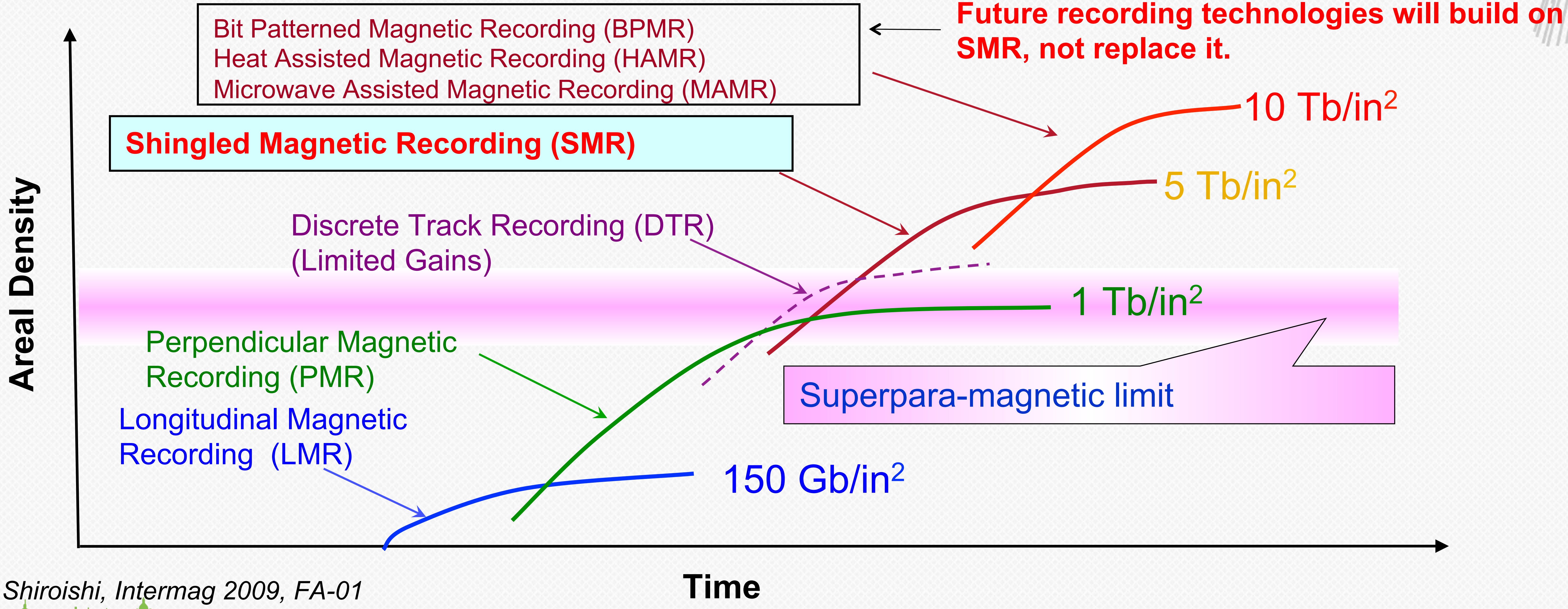
Jorge Campello
Director of Systems Architecture, HGST

OPEN
Compute Summit
Engineering Workshop

October 30-31, 2014
Paris

Magnetic Recording System Technologies

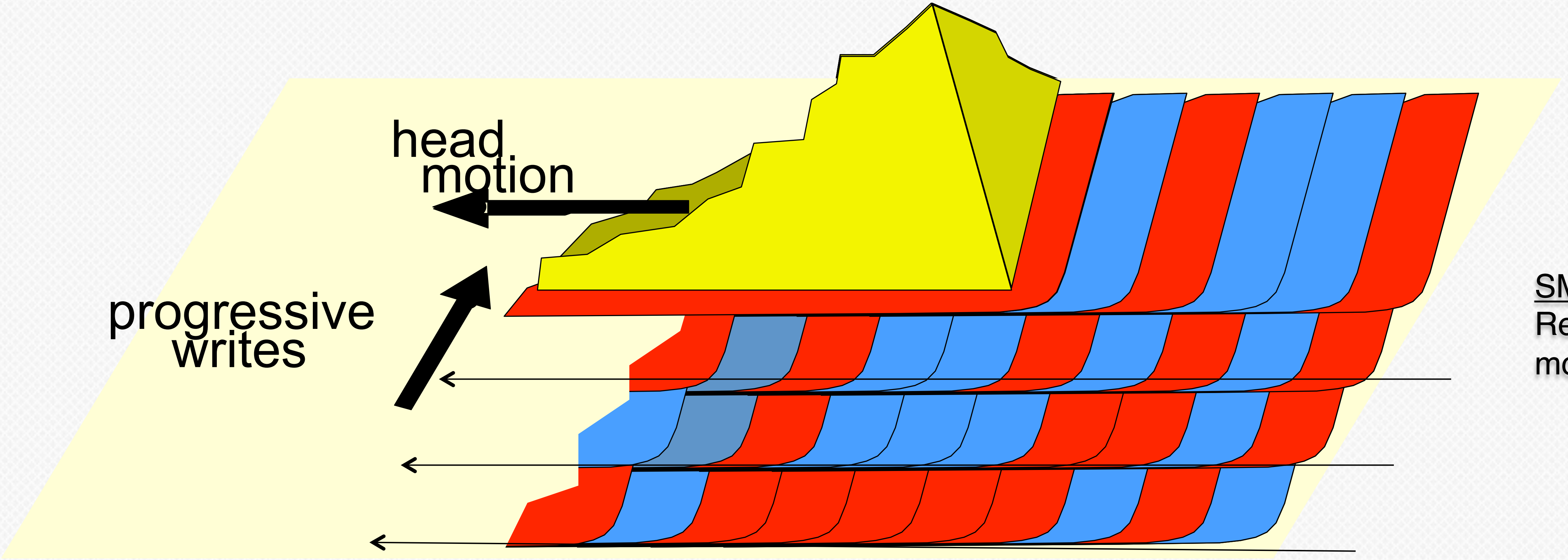
New recording system technologies are needed to keep the HDD industry on its historical track of delivering capacity improvements over time



Y. Shiroishi, Intermag 2009, FA-01

What is Shingled Magnetic Recording (SMR)?

SMR write head geometry extends well beyond the track pitch in order to generate the field necessary for recording. Tracks are written sequentially in an overlapping manner forming a pattern similar to shingles on a roof.



<u>SMR Constraint:</u>
Rewriting a given track will damage one or more subsequent tracks.

Wood, Williams, et al., IEEE TRANSACTIONS ON MAGNETICS, VOL. 45, NO. 2, FEBRUARY 2009

SMR Types

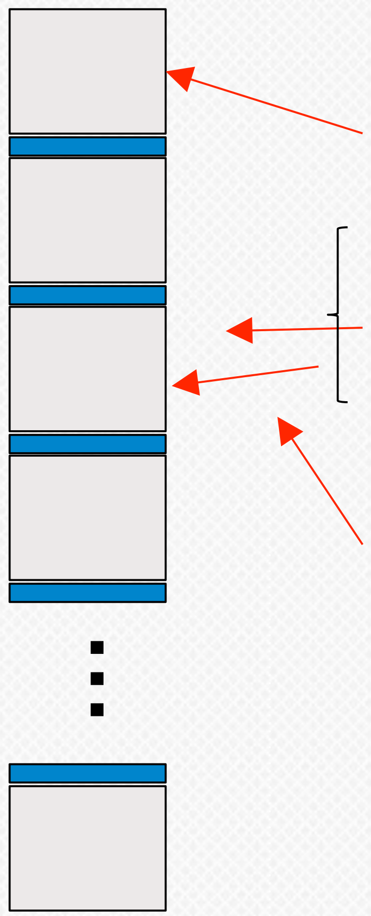
SMR category	Description
Drive managed (Autonomous)	No host changes. SMR device manages all requests. Performance is unpredictable in some workloads. Backward compatible
Host aware	Host uses new commands & information to optimize write behavior. If host sends sub-optimal requests the SMR device accepts the request but performance may become unpredictable. Backward compatible
Host Managed	Host uses new commands & information to optimize write behavior. Performance is predictable. If host sends sub-optimal requests the SMR device rejects the request. Not backward compatible

T10/T13 ZBC/ZAC

ZBC = Zoned Block Commands

ZAC = Zoned ATA Commands

Zoned Block Devices



Conventional Zone

Sequential Write Preferred Zone

Sequential Write Only Zone

Write Pointer

3 types of Zones supported

Conventional Zones

- Behave according to the direct access block device type model in SBC-3

Sequential Write Preferred Zones

- Implements the new ZBC standard
- Writes should be at the "Write Pointer" (WP) for best performance
  - BUT, Device will accept writes in any order

Sequential Write Only Zones

- Implements the new ZBC standard
- Writes have to be at the Write Pointer

Host Managed ← Two Device Types → **Host Aware**

Host Managed

- Sequential Write Only Zones; Conventional Zones are optional
- Reads cannot span zones or cross the Write Pointer

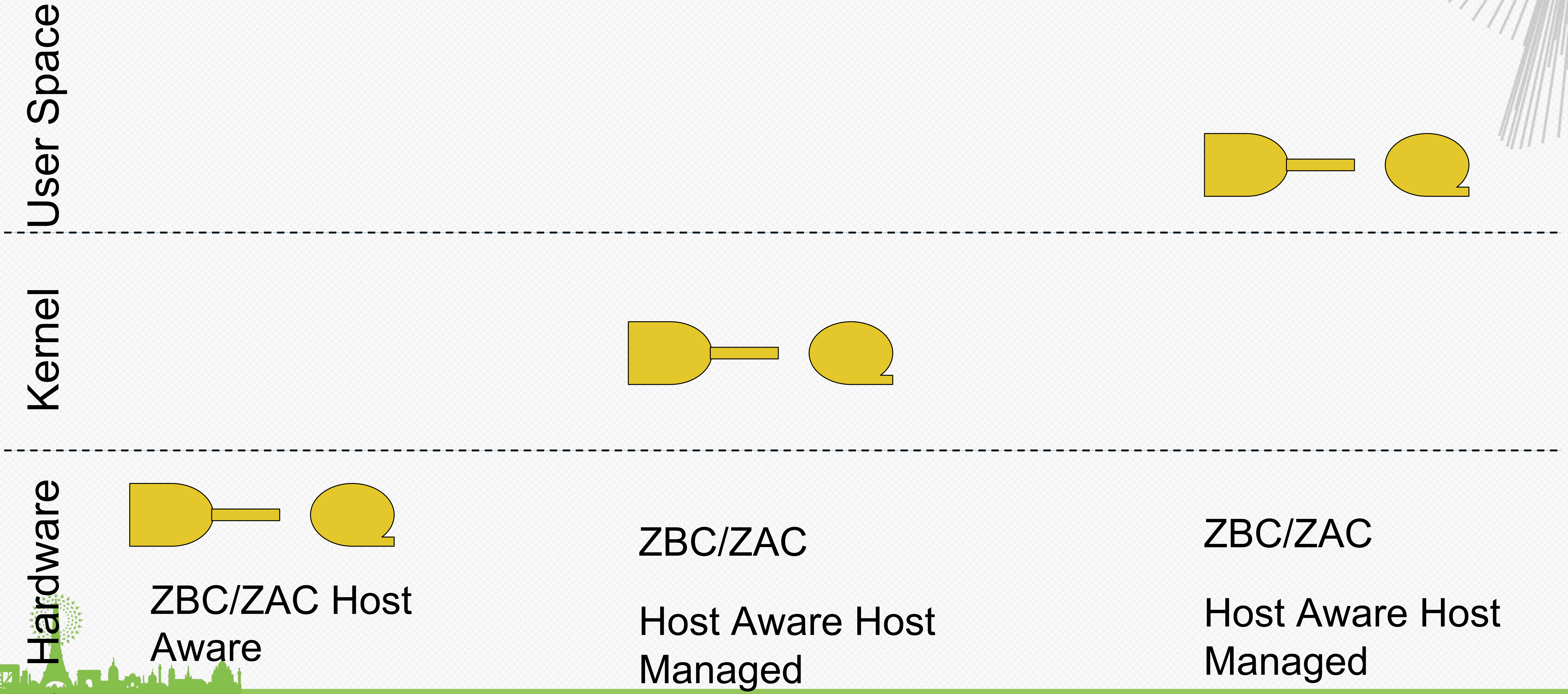
Host Aware

- Sequential Write Preferred Zones, Conventional Zones are optional
- Non-sequential writes in a Sequential Write Preferred Zone toggle the zone to conventional mode— dealt by HDD internal indirection

ZBC/ZAC Device Types – current drafts

	Direct Access	**Host Aware**	**Host Managed**
Peripheral Device Type	00h	00h	14h
HAW_ZBC	0b	1b	0b
Conventional zones	n/a	Optional	Optional
Seq'l wr preferred zones	n/a	**Mandatory**	Disallowed
Seq'l wr only zones	n/a	Disallowed	**Mandatory**
Reads and writes crossing seq'l write only zone boundaries	n/a	n/a	Disallowed
REPORT ZONES	Disallowed	Mandatory	Mandatory
RESET WRITE POINTER	Disallowed	Mandatory	Mandatory

SMR Introduction Models



User Space

Kernel

Hardware

ZBC/ZAC Host Aware

ZBC/ZAC

Host Aware Host Managed

ZBC/ZAC

Host Aware Host Managed

libzbc and lkvs:
Linux ZBC library and Linear Key Value Store Application

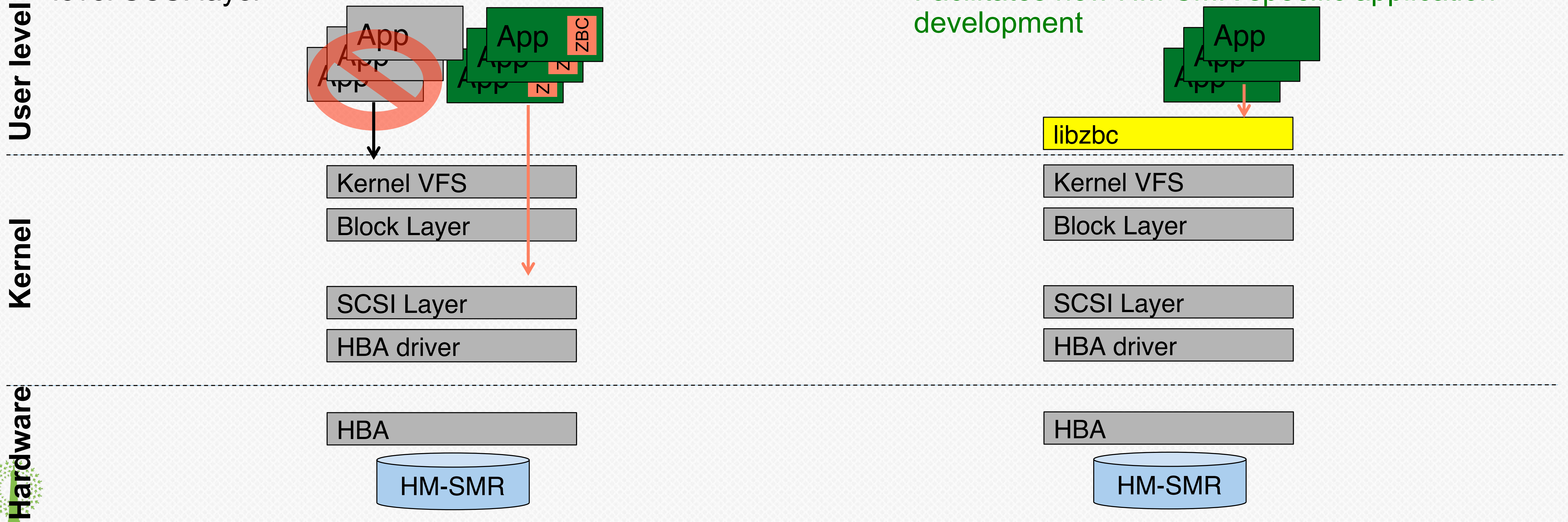
Designing for Host Managed SMR

Current State

- Existing applications will not work
- Need new applications that are HM-SMR specific
- Each app has to do their own ZBC parsing to talk with low level SCSI layer

HGST developed libzbc

- Removes annoyance of low-level parsing SCSI/ZBC commands
- Follows the T10/T13 standards
- Facilitates new HM-SMR specific application development



User level

Kernel

Hardware

App

App

App

App ZBC

App Z

App Z

Kernel VFS

Block Layer

SCSI Layer

HBA driver

HBA

HM-SMR

App

App

App

libzbc

Kernel VFS

Block Layer

SCSI Layer

HBA driver

HBA

HM-SMR

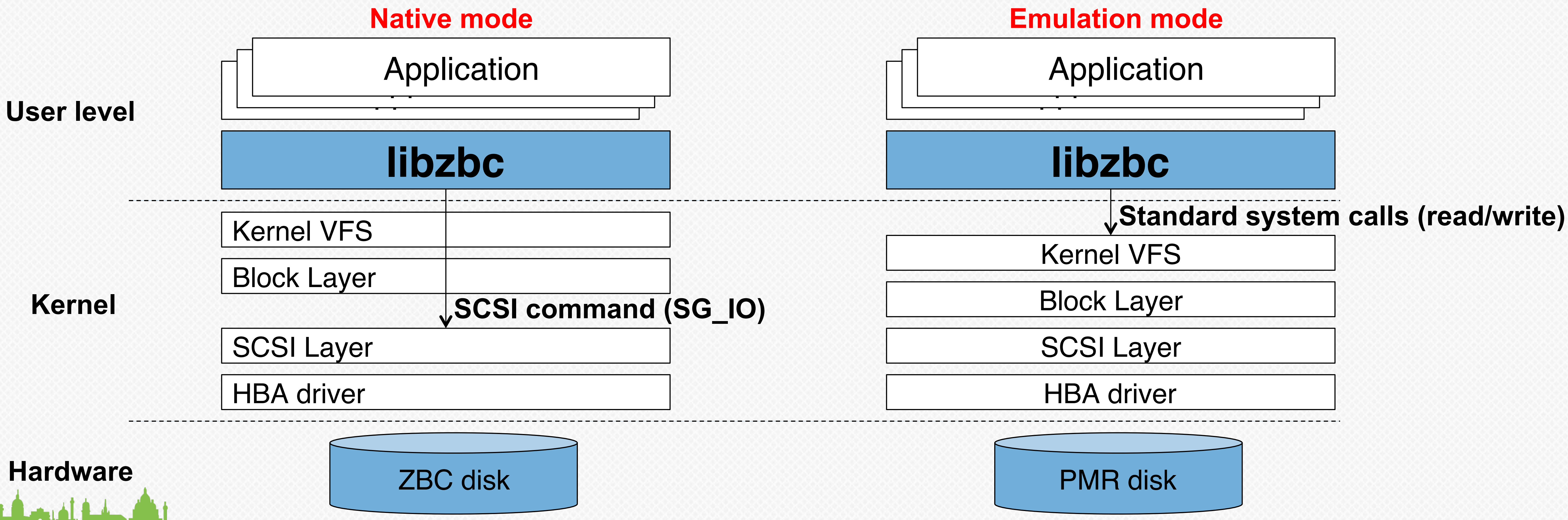
Libzbc Project: SMR for Linux

Download Now:
http://github.com/hgst

- Allows Linux apps access to host-managed HDD
- Ensures new command sets flow through HBA
- Emulates Host Managed SMR on PMR drives

libzbc

- Allows Linux applications access to ZBC host-managed disks
  - Access to disk zone information and read/write operations in zones through direct SCSI command execution (SG_IO)
  - ZAC drives will be supported by libzbc as well
- Additionally, provide a ZBC emulation layer for operation on top of standard SAS/SATA block devices
  - Zone configuration of the disk is emulated within the library



Libzbc Project

Libzbc is an Open Source Project

- Distributed under an LGPL licence

http://Github.com/hgst

The mailing list for the project is: libzbc@vger.kernel.org

Will provide a consistent interface for both ZBC and ZAC devices.

Will evolve with the standards

Currently supports ZBC Host Managed Devices or Emulation Mode

- Plan to support ZAC devices soon
- Plan to support Host Aware ZBC/ZAC as well

libzbc Interface

Functions	Description	Input	Output	SCSI command (native mode)
zbc_open	Open a device	Device file path	Device handle	INQUIRY, READ CAPACITY 16
zbc_close	Close an open device	Device handle	None	None
zbc_get_device_info	Get a device information (size, sector size, ...)	Device handle	Device information	None
zbc_report_zones	Get information on zones following a specified LBA	Device handle, zone start LBA, zone filter	Zone information	REPORT ZONES
zbc_reset_write_pointer	Reset the write pointer of an open or full zone	Device handle, zone start LBA	None	RESET WRITE POINTER
zbc_pread	Read data from a zone	Device handle, Zone to read, LBA offset in the zone, number of sectors to read, data buffer	Amount of sectors read and data	READ 16
zbc_pwrite	Write data to a zone	Device handle, Zone to write, LBA offset in the zone, number of sectors to write, data buffer	Amount of sectors written	WRITE 16

libzbc Interface (Emulation Mode)

- These functions are used to initialize an emulated ZBC device
  - Write pointer persistency is also emulated
    - Zone configuration and current write pointer values are saved to the disk on execution of the zbc_close function

Functions	**Description**	**Input**	**Output**	**SCSI command (native mode)**
zbc_set_zones	Configure the zones of an emulated device	Device handle, size of conventional zone, size of sequential write zones	None	None*
zbc_set_write_pointer	Change a zone write pointer LBA value	Device handle, zone start LBA, write pointer value	None	None*

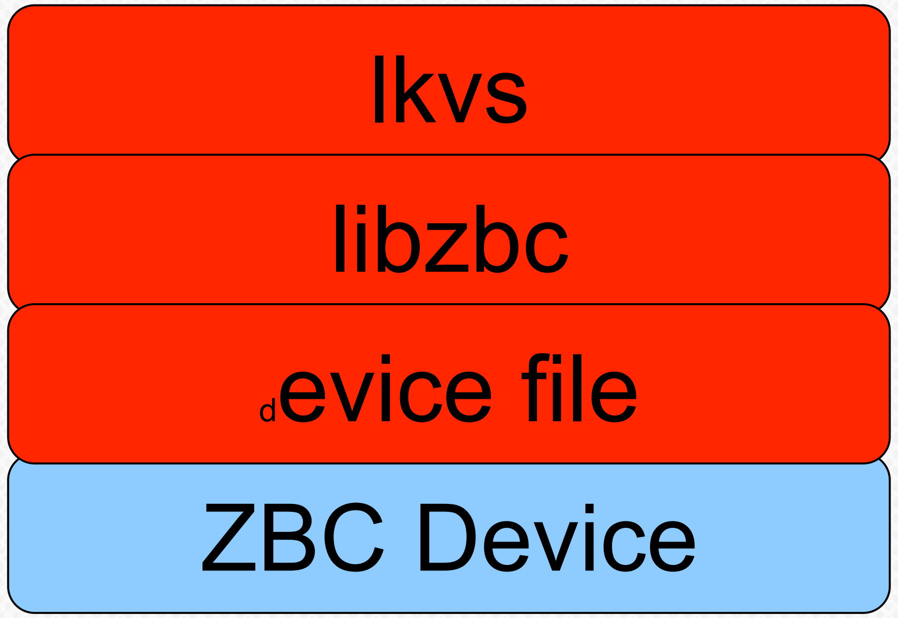
Linear Key Value Store (lkvs) Application

Linear Key Value Store Architecture

- lkvs
  - Implements a simple append only KVS as an example use of libzbc
  - Queries drive info (write pointer, zone information) through libzbc
  - Read/write executed through libzbc
- libzbc
  - Provides zone information, write pointers, to lkvs

Applications link with libzbc

lkvs gets ZBC device information and read/write operations are perfromed through libzbc



Ikvs Interface

Functions	Description	Input	Output
openDev	Open a device	Device file path, format flags	Bool success
Put	Insert key/value pair into the store	Key string, value buffer, size	Bool success
Get	Get key/value pair form the store	Key string, value buffer, size	Bool success
List	List key/value pairs on the device (Not Finalized)	TBD	TBD

OPEN
Compute Summit

Engineering Workshop

October 30-31, 2014
Paris

Thank YOU!