SNIA ESF | ETHERNET STORAGE

EVERYTHING YOU WANTED TO KNOW ABOUT STORAGE, BUT WERE TOO PROUD TO ASK:

Part Vermillion

What if Programming and Networking Had a Storage Baby?

July 6, 2017
10:00 am PT

SNIA Legal Notice

SNIA ESF | ETHERNET STORAGE

- The material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Today's Presenters

Alex McDonald
NetApp

Dror Goldenberg
Mellanox

Rob Peglar
Symbolic IO

J Metz
Cisco

SNIA-at-a-Glance

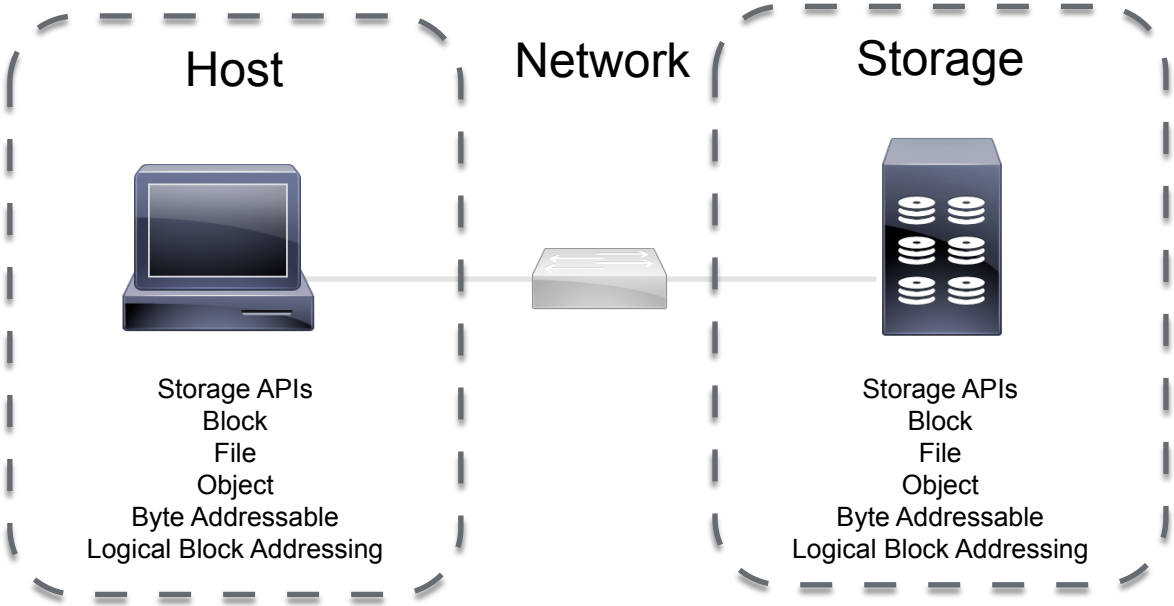
160
unique member companies

2,500
active contributing members

50,000
IT end users & storage pros worldwide

Agenda

- Block vs. File vs. Object
- Byte Addressable vs. Logical Block Addressing
- POSIX and Storage
- Log Structures, Journaling Systems



Block vs. File vs. Object

Block, File and Object Storage

- Three types of storage access
  - Block
  - File
  - Object
- Each has distinct characteristics to relationships with hosts
  - Distinct advantages/disadvantages
- These are not the same thing as File Systems!

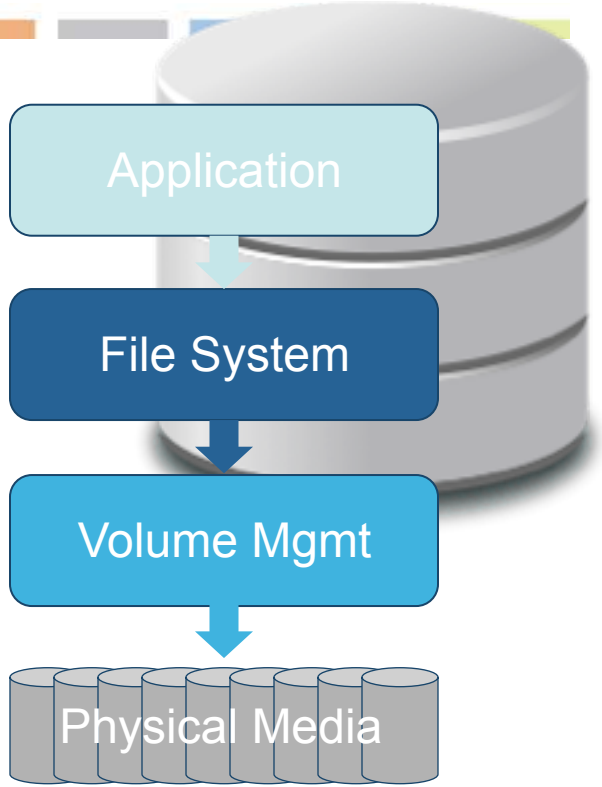
Block, File and Object Storage

- Block
  - The unit in which data is stored and retrieved on disk and tape devices; the atomic unit of data.

Source: SNIA Dictionary

Block Storage

- What can you do with it?
  - Boot servers/VMs
  - Works very well with databases and transactions
- Pros
  - Host system has direct access to storage memory (drives, disk, NVM)
  - Highest performance capabilities
- Cons
  - Heavy reliance on HA redundancy at every level of the architecture



Block, File and Object Storage

- File
  - An abstract data object made up of (a.) an ordered sequence of data bytes stored on a disk or tape, (b.) a symbolic name by which the object can be uniquely identified, and (c.) a set of properties, such as ownership and access permissions that allow the object to be managed by a file system or backup manager

Source: SNIA Dictionary

File Storage

- Foundation for Network Attached Storage (NAS)
  - **NFS, SMB**
  - CIFS (deprecated)
- Enterprise
  - Purpose-built, structured and unstructured data over one or more protocols
  - Scalable and higher performance
  - Supports large number of clients
  - Features: tiering, caching, de-duplication, multi-tenancy, replication, multi-protocol support, etc.
  - Suited for large data sets, data sharing
  - Clustered NAS (Scale-up or scale-out)
  - Petabyte scale, 1000s of drives

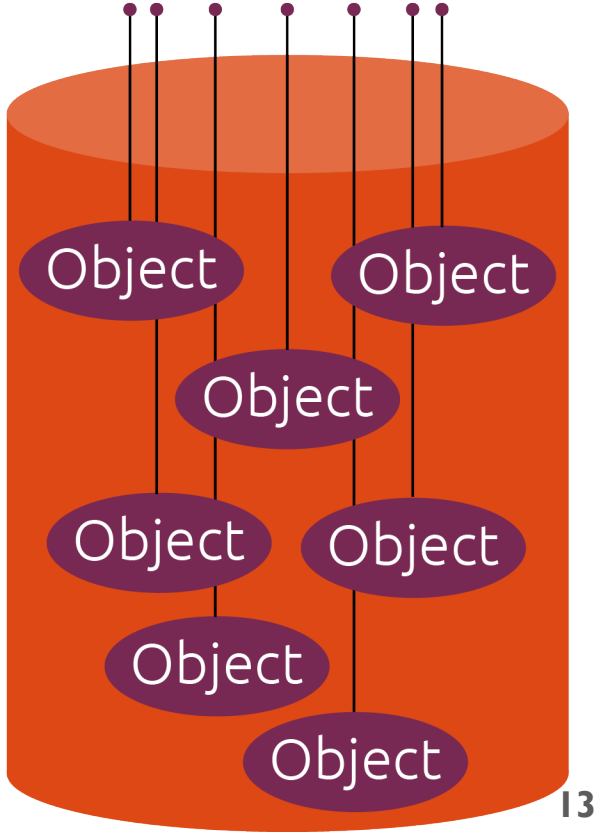
Block, File and Object Storage

- Object
  - The encapsulation of data and associated metadata

Source: SNIA Dictionary

Object Storage

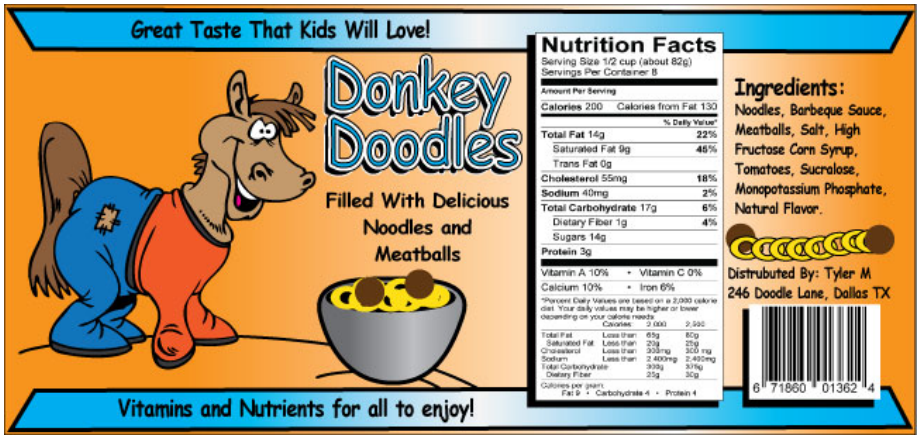
- Instead of a hierarchy, object-storage organizes things in a flat structure. This allows for massive scalability.
- Though it has no file system, like file storage, changes are at the file level.
- Instead of a file system, objects have lots of metadata. These attributes can be built-in or customer-defined.



Visualizing Object Storage

- Imagine a grocery store with no labels on any of the cans

- Metadata is the information on the label of the cans



- Over time, metadata can be more important than the data itself

Concept borrowed shamelessly from Jeff Lundberg, HDS "The Fundamentals of Object Storage, Part 1", because it was so brilliant.

Unique Abilities of Object

- Enables the user to find data based upon Regular Expressions
  - You can search in very large datasets on metadata
  - Object storage is ideal for **analytics** applications
- Allows you to treat the Cloud not as a large "Object Store" but as a database
- As the size of the Cloud grows, so does your ability to find data
  - The better your metadata is, the better your queries can be
- Examples of using complex queries:
  - Find objects of a certain age and containing specific meta-data
  - Find objects belonging to a person or application that can be removed but including other criteria for exclusion
  - Find similar objects and classify them
- Recall: SCSI doesn't have the ability to "find objects"

Use Cases

Block

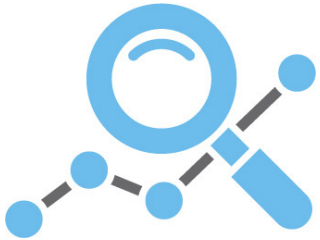
- Databases
- Transactional Processing
- Enterprise-wide Applications
- Dedicated Network/High Performance

File

- Departmental Applications
- End User Data/ Files
- Shared/Clustered Systems

Object

- Analytics
- Unstructured Data Applications
- Cost effective
- Extremely scalable



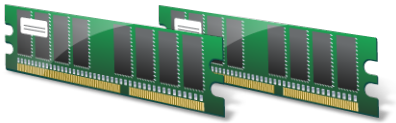
Side x Side Comparison

	Block	File	Object
Transaction Units	Blocks	Files	Objects, that is, files w/ custom metadata
Supported Update Types	Supports in-place updates	Supports in-place updates	No in-place update support; updates create new object versions
Protocols	SCSI, Fibre Channel, SATA. NVMe	SMB and NFS	REST and SOAP over HTTP
Metadata Support	Fixed system attributes	Fixed file-system attributes	Supports custom metadata
Best suited for	Transactional, and frequently-changing data	Shared file data	Relatively static file data and cloud storage
Biggest Strength	High performance	Simplified access and management of shared files	Scalability and distributed access
Limitations	Difficult to extend beyond the Data Center	Difficult to extend beyond the Data Center	Ill-suited for frequently changing transactional data, doesn't provide a sharing protocol with a locking mechanism

Byte Addressable vs. Logical Block Addressing

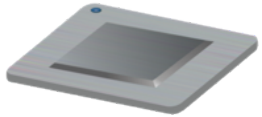
System Architecture

DRAM



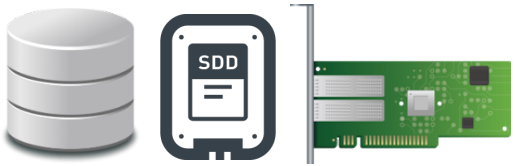
Memory Bus

CPU



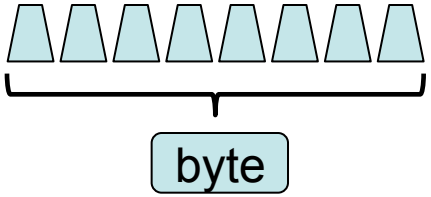
PCIe Bus

Peripherals: Storage & Network

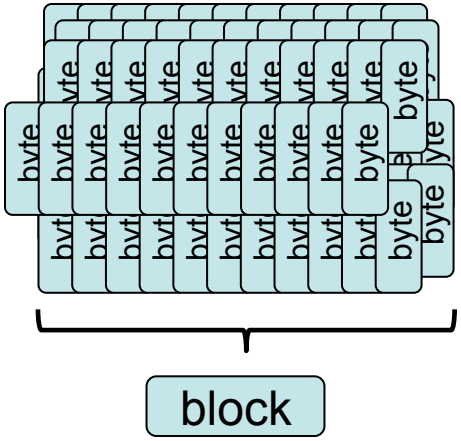


Bit=1 or 0

Byte = 8 bits



Block = 512 bytes, or 1024, 4096, 32K, 4 million bytes, etc.



Addressing

Address - describes the location

What is byte/block addressing?

Byte and Block Addressing

- Byte Addressable: Access one byte at a time
  - 1 byte = 8 bits
  - Traditionally used for memory (volatile)
- Logical Block Addressable
  - Access an entire block (many bytes) at a time
  - Traditionally used for storage (persistent)
  - Typical access is 512 bytes (sector)

Comparing Byte vs. Logical Block

- Byte Addressable
  - Fine grained access
  - 8 bits at a time
  - Load/store commands
  - 32-bit address → 4GB of memory
  - Traditional on memory bus
- LBA
  - Coarse access
  - Often 512-byte blocks
  - I/O commands
  - 32-bit address → 2TB of storage.
  - Traditional on storage dev

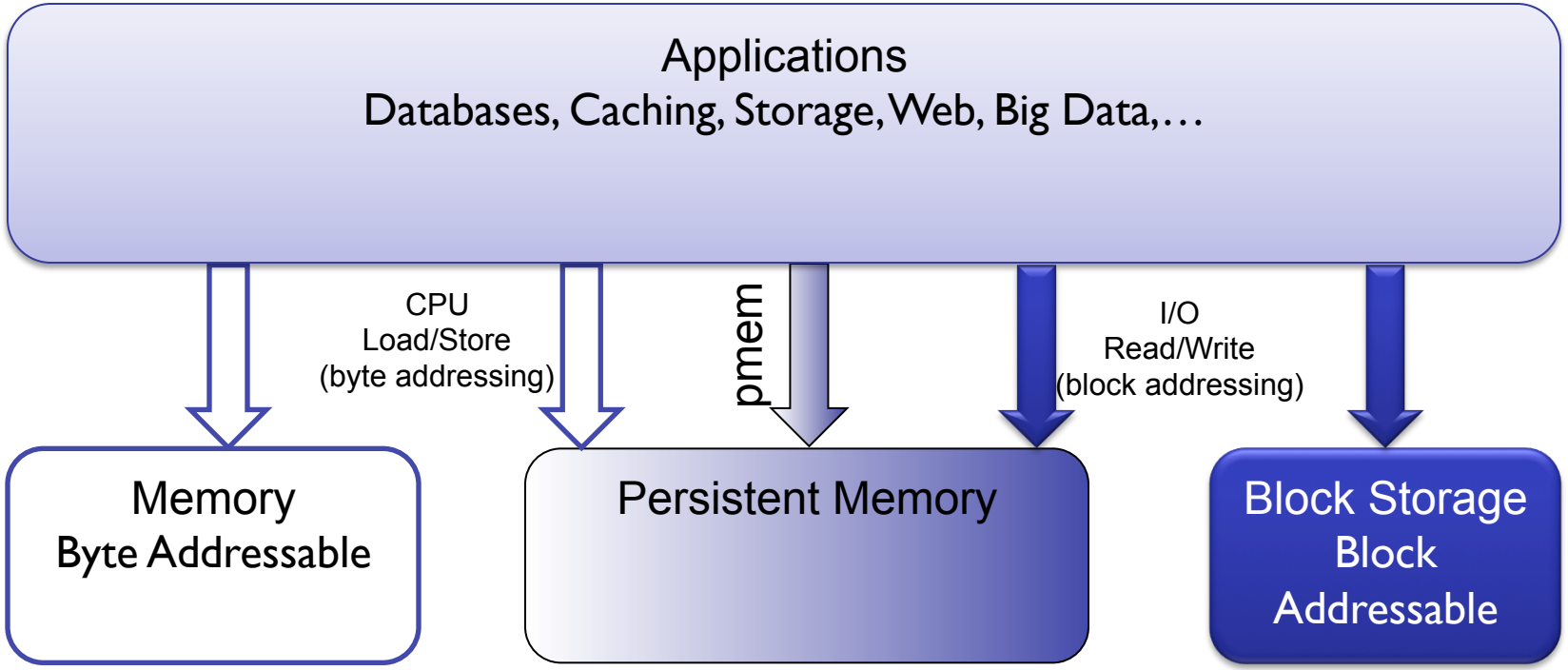
What About Persistent Memory?

- Traditionally…
  - Memory is fast and volatile on memory bus
  - Storage is slow and persistent on peripheral bus
- New persistent memory breaks the rules
  - Fast like memory, persistent like storage
  - Fast storage on PCIe bus, or…
  - Persistent memory on memory bus
- New access and programming models

NVM Programmable Models

- Access options
  - Place on DRAM bus but use LBA like storage
  - Place on DRAM bus and address like memory
  - Place on DRAM bus and address like storage class memory
- Need to deal with persistency
- See SNIA NVM Programming Model
  - https://www.snia.org/tech_activities/standards/curr_standards/npm

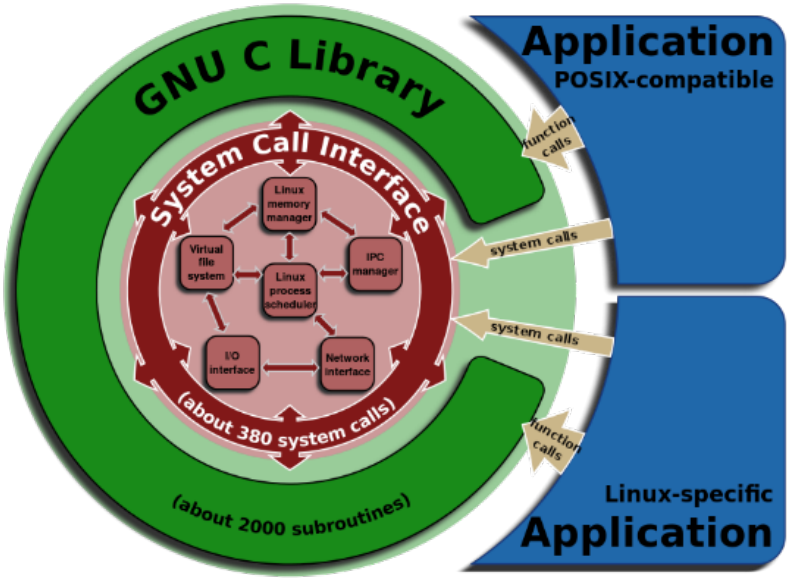
Summary



POSIX and Storage

POSIX & Storage

- POSIX
  - “Portable Operating System Interface for uniX”
  - Goal; provide portability for applications across multiple OSes
  - Family of standards, specified by the IEEE
    - Late 1980s thru 1997; see http://standards.ieee.org/develop/wg/POSIX.html
    - Includes C programming language standard
  - Specifies API level interfaces for a wide variety of operating system interfaces



POSIX & Storage

- Storage related parts
- POSIX layer lives above device driver (or block level)
  - Provides directories and files
  - Data as stream of bytes
- Sets of I/O operations
  - open, close, read, write, lseek, ioctl
  - Plus a large number of ancillary calls and CLI cmds in support

• open	• fsetpos	• fchownat
• read	• fclose	• faccessat
• write	• fsync	• utime
• close	• creat	• futimes
• lseek	• readdir	• lutimes
• llseek	• opendir	• futimesat
• _llseek	• fopendir	• link
• lseek64	• rewinddir	• linkat
• stat	• scandir	• unlinkat
• fstat	• seekdir	• symlink
• stat64	• telldir	• symlinkat
• chmod	• flock	• rmdir
• fchmod	• lockf	• mkdirat
• access	• lseekm	• getxattr
• rename	• lstat	• lgetxattr
• mkdir	• fstatat	• fgetxattr
• getdents	• fopen	• xetxattr
• fcntl	• fdopen	• lsetxattr
• unlink	• freopen	• fsetxattr
• fseek	• remove	• listxattr
• rewind	• chown	• llistxattr
• ftell	• fchown	• flistxattr
• fgetpos	• fchmodat	• removexattr

POSIX & Storage

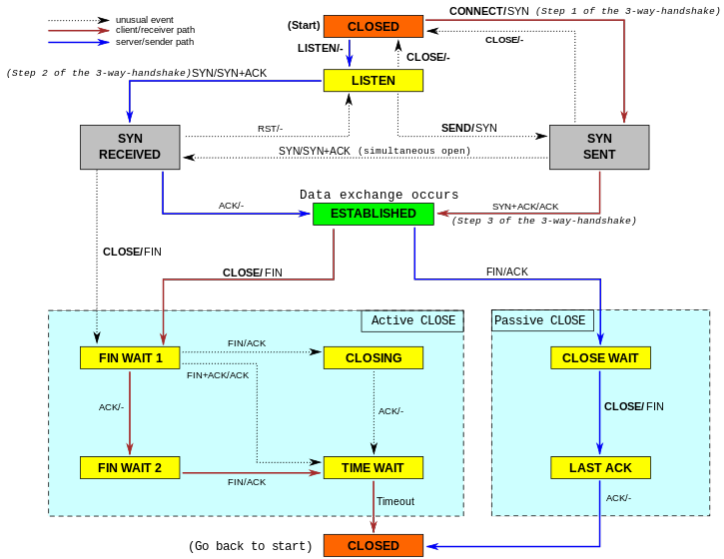
- Benefits of POSIX
  - Simplicity & broad application
    - Data as a stream of bytes matches capabilities of wide range of devices
  - Consistency & locking
    - Last writer wins, read gets latest written, file & byte range locking
  - Easy[1] to port apps across POSIX compliant systems
  - Provides state through an opaque[2] file handle
  - File systems are ubiquitous (and there a huge variety of them) but most have similar to identical POSIX interfaces

European Commission

[1] For some definition of “easy” that doesn’t include “impossible”
[2] Or transparent depending on your view of the semantics of these words

POSIX & Storage

- Downsides of POSIX
  - Definitions are implemented as a C language API
  - Stateful
    - Not RESTful
    - Compare: HTTP provides POST, GET, PUT, DELETE; no handle, just resource IDs in the form of URLs, and is idempotent with no locking
  - Network unfriendly
    - Chatty; network latency an issue; every operation requires client to server acknowledgements; locking
    - Difficult & complex recovery in the face of bad network connectivity
  - Metadata, operation ordering & cache coherence…



POSIX & Storage

- Positive future for POSIX
  - File system protocols are maturing & becoming more capable
    - Protocols allow talking to file systems over a network
    - NFSv4.x, SMB3.x (“POSIX-like” semantics)
  - Interesting file system developments
    - High-performance clustered file systems; Btrfs, Ceph, GlusterFS and more
    - File systems in IoT edge devices
  - Cloud & object type stores can’t (yet) replace all the functionality apps require
    - Transactional systems
    - Stream oriented data increasingly important

Log Structuring and Journaling

and other fun stuff

Log Structured Systems

Log Structured Systems

- Definition – a system where all incoming metadata and data are written sequentially to a circular buffer, called a log
  - First proposed in 1988 (Ousterhout and Douglis)
  - First implemented in Sprite (Unix-like distributed OS) by Ousterhout and Rosenblum in 1992
- Non-log systems write randomly, overwrite-in-place
- Log systems ‘batch up’ updates and write sequentially
  - Not overwrite-in-place
  - Log enables crash recovery (end of log), checkpoints (last known good persist), roll-forward

Basic Operation

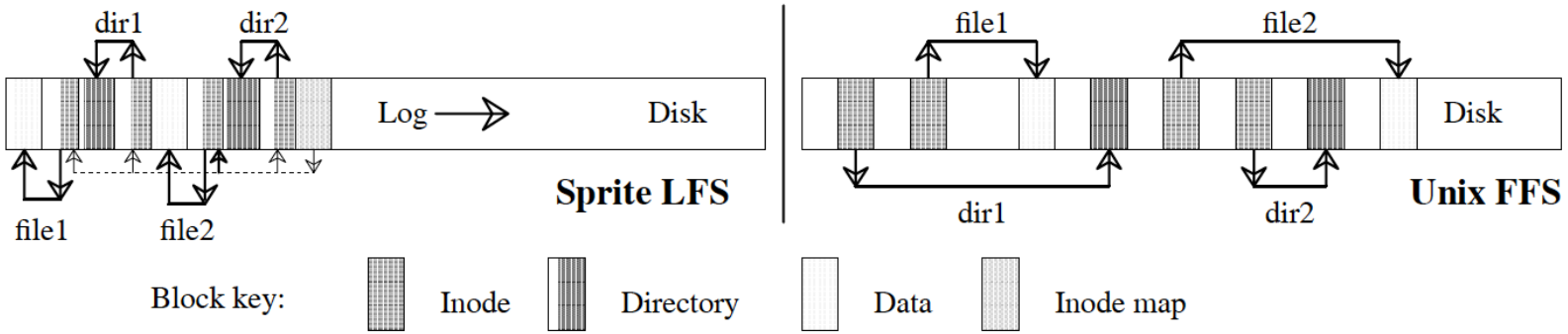
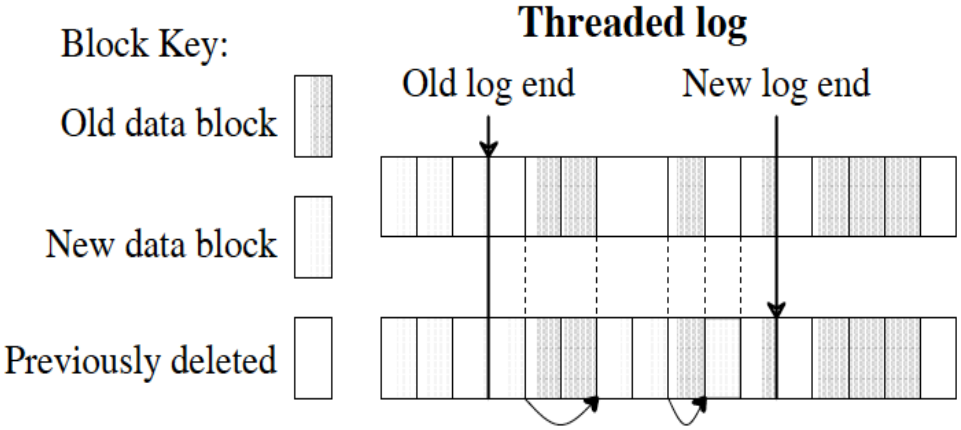


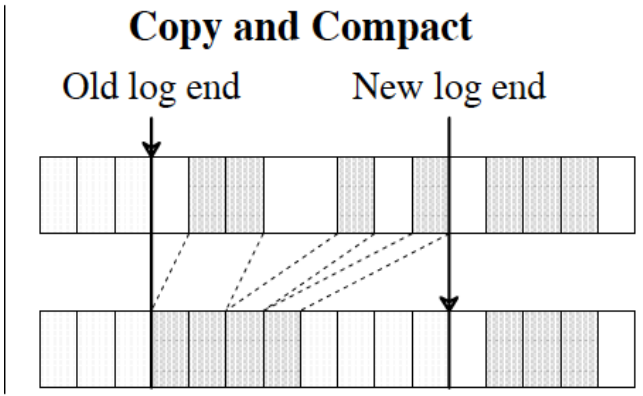
Figure 1 — A comparison between Sprite LFS and Unix FFS.
This example shows the modified disk blocks written by Sprite LFS and Unix FFS when creating two single-block files named `dir1/file1` and `dir2/file2`. Each system must write new data blocks and inodes for `file1` and `file2`, plus new data blocks and inodes for the containing directories. Unix FFS requires ten non-sequential writes for the new information (the inodes for the new files are each written twice to ease recovery from crashes), while Sprite LFS performs the operations in a single large write. The same number of disk accesses will be required to read the files in the two systems. Sprite LFS also writes out new inode map blocks to record the new inode locations.

Attribution: "The Design and Implementation of a Log-Structured File System"
Ousterhout and Rosenblum, July 1991

Segment Cleaning (aka GC)



Skip over active blocks, overwrite deleted/overwritten blocks



Rewrite active blocks into new space, move end

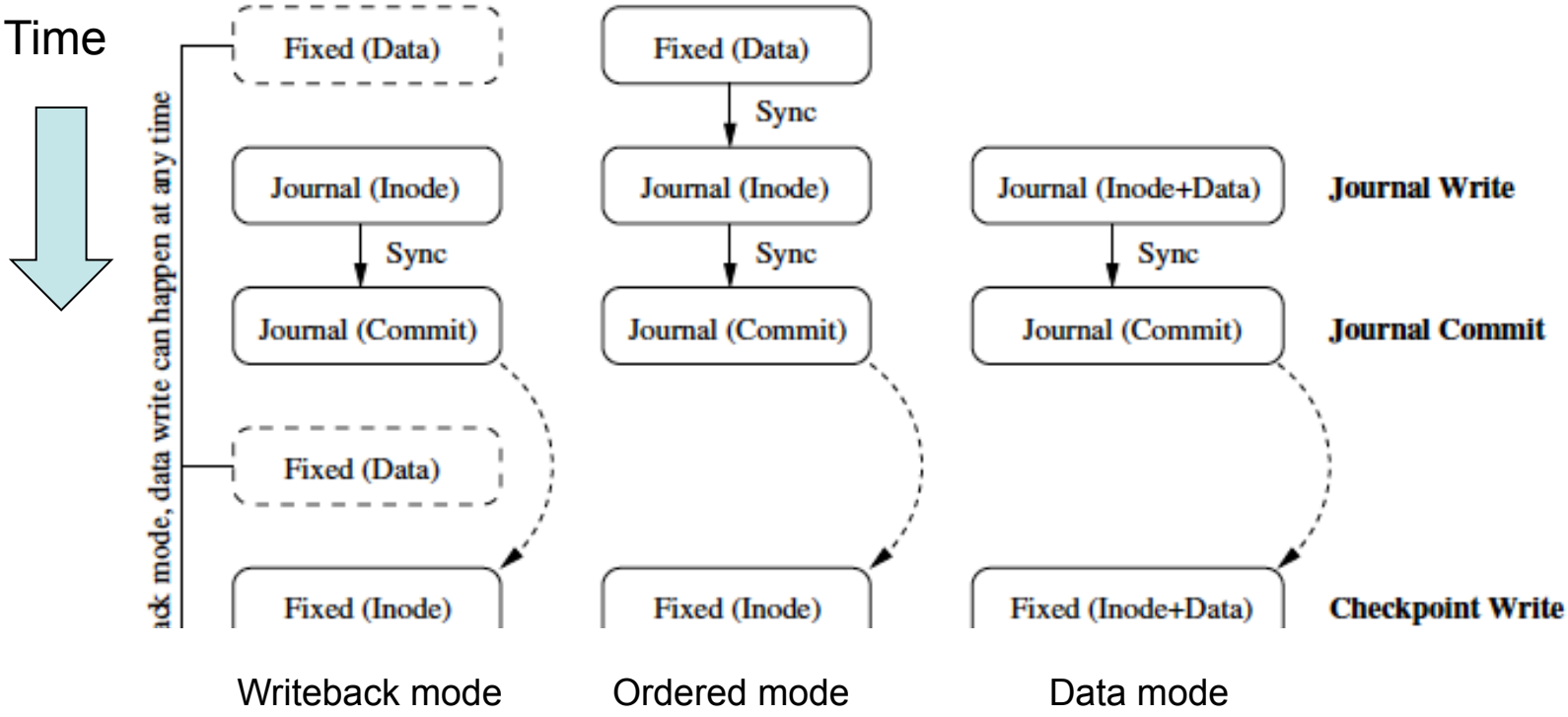
Attribution: “The Design and Implementation of a Log-Structured File System”
Ousterhout and Rosenblum, July 1991

Journaling Systems

Journaling Systems

- Definition – a system where intended metadata for writes (updates) are recorded in a journal - but not necessarily data
  - Very popular – first implemented in 1990 (JFS)
  - Many systems today – NTFS, ext3, ext4, ReiserFS, etc., including split-journal
- Record pending persistence – then commit (aka 2-phase, write twice)
- Provides for true atomicity – known persist success (or failure)
- Comparison:
  - Log-Structured –is- the filesystem; journal is only ½ the filesystem
    - Journal has crash recovery (journal replay), assured last-known-good, optional barriers

Basic Operation (ext3 example)



Attribution: "Analysis and Evolution of Journaling File Systems" Prabhakaran and Arpaci-Dusseau, USENIX 2005

Log Structuring and Journaling

Wasn't that fun?

Summary

- Block/File/Object
  - Fundamental storage types in Data Center applications, each requiring performance/ flexibility trade-offs
- Byte Addressable v. Logical Block Addressable
  - Traditional ways to access memory and storage respective, whose use is changing with the adoption of persistent memory
- POSIX and APIs
  - POSIX a standard that enables storage to be viewed as both a random access and a stream oriented source of data
- Log Structuring and Journaling
  - Two related techniques, that enhance consistency and recovery aspects of systems which persist data, especially filesystems

Other Storage Terms Got Your Pride? This is a Series!

SNIA ESF | ETHERNET STORAGE

Don't miss our next one – August 1, 2017

Everything You Wanted To Know About Storage But Were Too Proud To Ask – Turquoise

(Where Does My Data Go?)

- Volatile v Non-Volatile v Persistent Memory
- NVDIMM v RAM v DRAM v SLC v MLC v TLC v NAND v 3D NAND v Flash v SSDs v NVMe
- NVMe
- **Register at: https://www.brighttalk.com/webcast/663/267327**

Other Storage Terms Got Your Pride? This is a Series!

SNIA ESF | ETHERNET STORAGE

- Check out previously recorded webcasts:
- **Teal** – Buffers, Queues and Caches https://www.brighttalk.com/webcast/663/241275
- **Rosé** - All things iSCSI https://www.brighttalk.com/webcast/663/244049
- **Chartreuse** – The Basics: Initiator, Target, Storage Controller, RAID, Volume Manager and more https://www.brighttalk.com/webcast/663/215131
- **Mauve** – Architecture: Channel vs. Bus, Control Plane vs. Data Plane, Fabric vs. Network https://www.brighttalk.com/webcast/663/225777
- **Sepia** – Getting from Here to There https://www.brighttalk.com/webcast/663/249431

Speaking of Series…Check out Storage Performance Benchmarking

SNIA ESF | ETHERNET STORAGE

- Storage Performance Benchmarking:
  1. Introduction and Fundamentals
  2. Solution under Test
  3. Block Components
  4. File Components

Watch them all on-demand at:
http://www.snia.org/forums/esf/knowledge/webcasts-topics

After This Webcast

SNIA ESF | ETHERNET STORAGE

- Please rate this webcast and provide us with feedback
- This webcast and a PDF of the slides will be posted to the SNIA Ethernet Storage Forum (ESF) website and available on-demand at www.snia.org/forums/esf/knowledge/webcasts
- A full Q&A from this webcast, including answers to questions we couldn't get to today, will be posted to the SNIA-ESF blog: sniaesfblog.org
- Follow us on Twitter @SNIAESF
- Need help with all these terms? Download the SNIA Dictionary http://www.snia.org/education/dictionary