EMC²
where information lives®

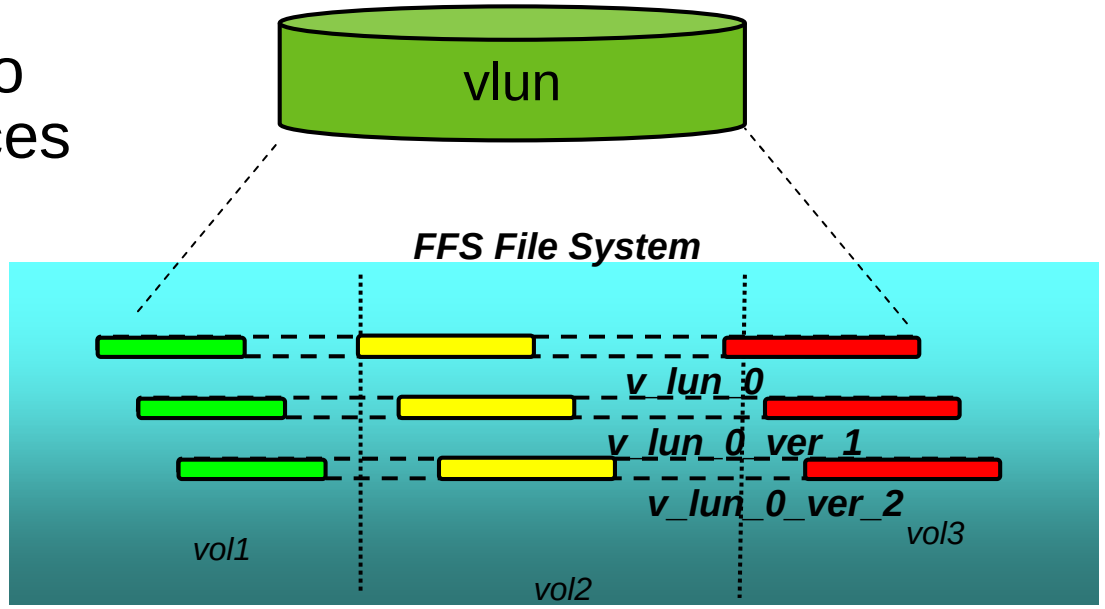
Storage Virtualization using a Block-Device File System

S. Faibish, S. Fridella, P. Bixby, U. Gupta – EMC

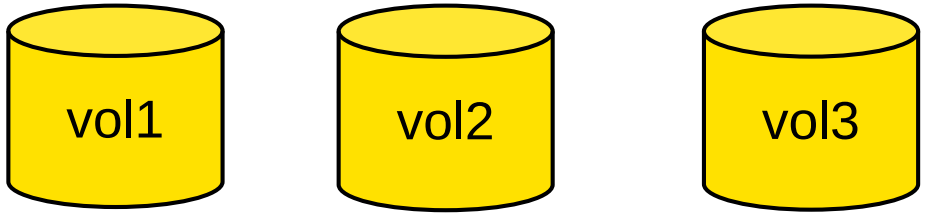
SYSTOR 2007, October 29, Haifa, Israel

Storage Virtualization; what is it?

- Automatic mapping of pieces of information to the ideal storage devices in a transparent and durable manner



- Virtualizes Storage Logical Units (LUNs) as files in a File System



Storage Virtualization; what is it? – (cont.)

- Achieve features that are not characteristic to typical files in file systems but specific to LUs such as:
  - Thin Provisioning for optimal capacity utilization
  - Location Independence and multiple copies for protection
  - Dynamic Allocation
  - Dynamic Re-mapping for Snapshots
  - Data location based on SLO for performance and reliability
  - Better power utilization efficiency
- Use techniques such as file versioning
- Best achieved using special Block-Device File System – a modified version of FFS

Why Use File Systems?

- Can achieve different SLO for files data and metadata
- SLO is related to information security, reliability and performance
- Information is stored in files making the file system the place to build SLO
- But traditional file systems can only achieve limited SLO:
  - UNIX like file systems: ext3, sVfs and FFS
  - Log structured file systems: LFS and WAFL
  - Windows based file systems: NTFS
- What about ZFS? Does it virtualize storage?

File System Historical Background

- Original file systems design assumptions
  - Disks are good at bulk sequential transfers of data.
  - Disks perform poorly when forced to seek.
  - Logically contiguous disk addresses are likely to be physically contiguous.
- Managing data is easy; managing metadata is hard
  - Meta-data objects are small, fragmented, and frequently accessed.
  - Meta-data operations have a very high seek-to-byte-of-data transferred ratio
  - Have generally poor performance if compared to data operations.
- Special techniques were used to manipulate the metadata
  - Collocate inodes close to directories and file data to reduce seeks
  - Organize data in fixed structures on disk such as Cylinder Groups (FFS) or Segments (LFS)
  - Use soft updates and explicit grouping for ordering metadata and data

What File System?

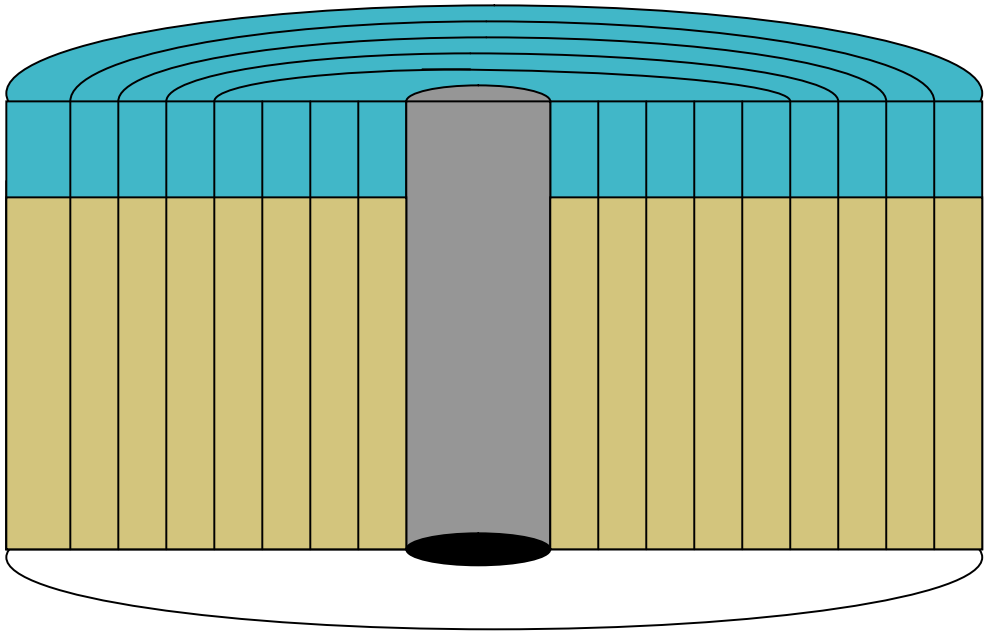
- Modern file systems don't rely on metadata and data co-locality; traditional approach is less beneficial
- Old/New file system concept based on the Berkeley Fast File System (FFS) splitting metadata and data because:
  - FFS significantly improved the Unix file system performance by spreading file system metadata across the disk.
  - The metadata is "stuff" used to control and maintain data, but not the data itself.
  - By grouping file data with its associated metadata, reference locality improved dramatically.
  - The specific layout on the disk was based on the physical characteristics of the disk and efforts were made to store related information the same physical disk cylinders.
- FFS allows easy implementation of file versioning

Many kinds of Metadata

- File system metadata and other kinds of metadata
- UxFS (EMC version of FFS) has both fixed and dynamic metadata using logical disks organized in Cylinder Groups
- Fixed Metadata – can be easily separated
  - Superblock and alternate superblocks
  - File inodes
  - Free/In-use bitmaps for block and inodes
  - Cylinder group headers with CG info including the bitmaps
- Dynamic Metadata
  - Indirect blocks – needs structural change
  - Directory blocks – can be easier to separate (ST, OSD, OSS)
- Other storage metadata
  - Parity and checksums
  - Encryption keys

Normal UxFS Layout on Disk

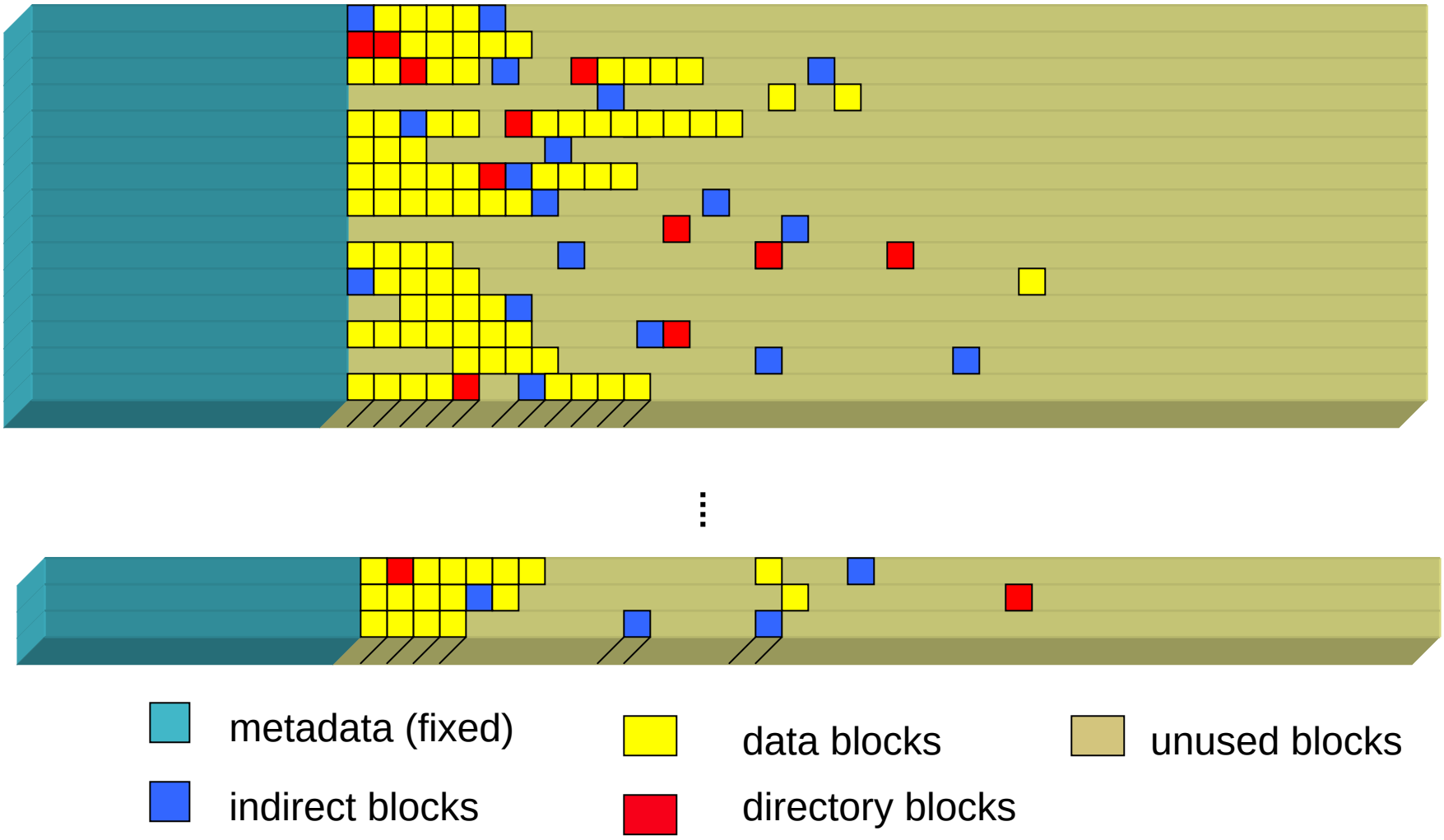
Each of the cylinder groups are (generally) the same size and shape. This simplifies extending the file system.



fixed metadata

data blocks and dynamic metadata

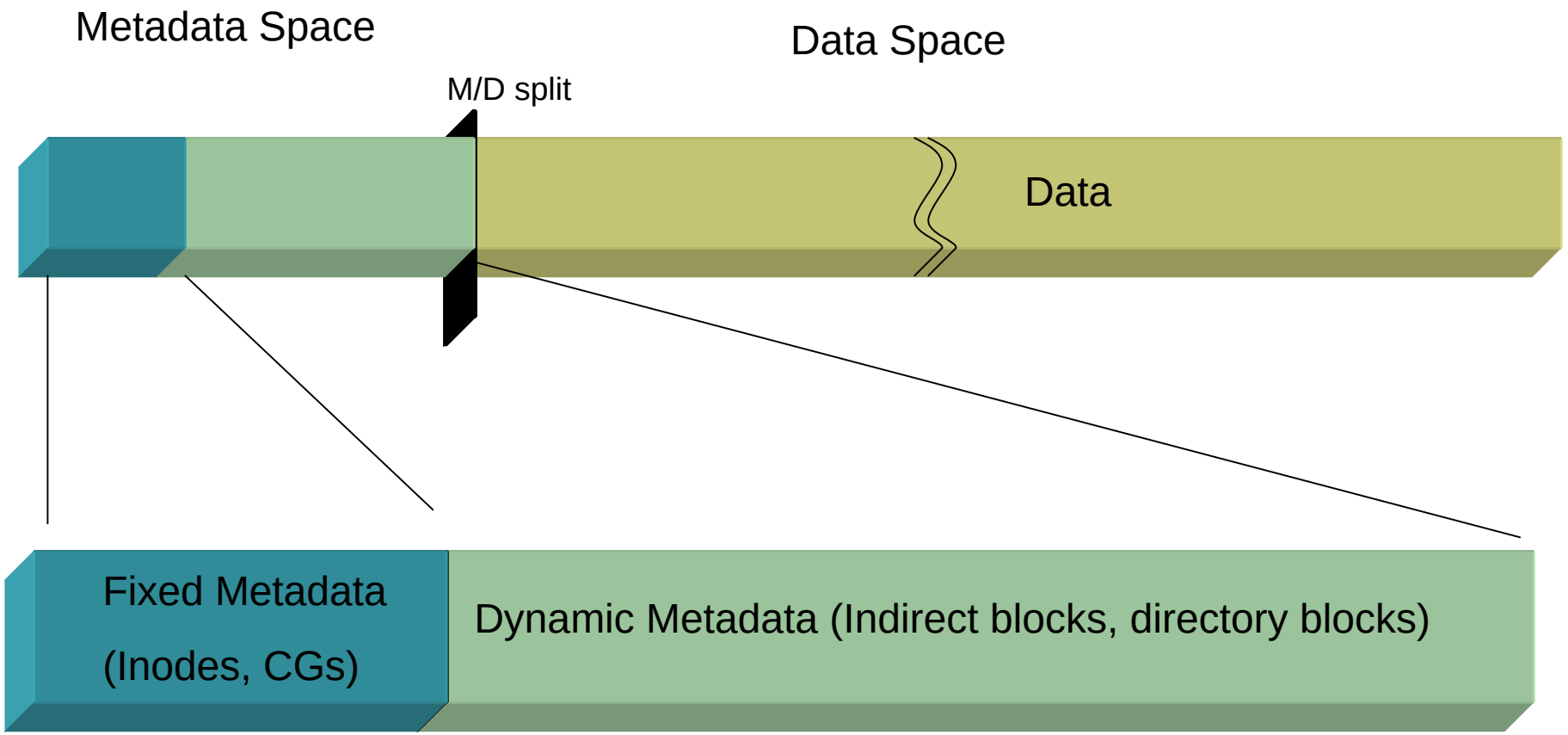
Normal UxFS Layout – w/ Indirect and Directory Blocks



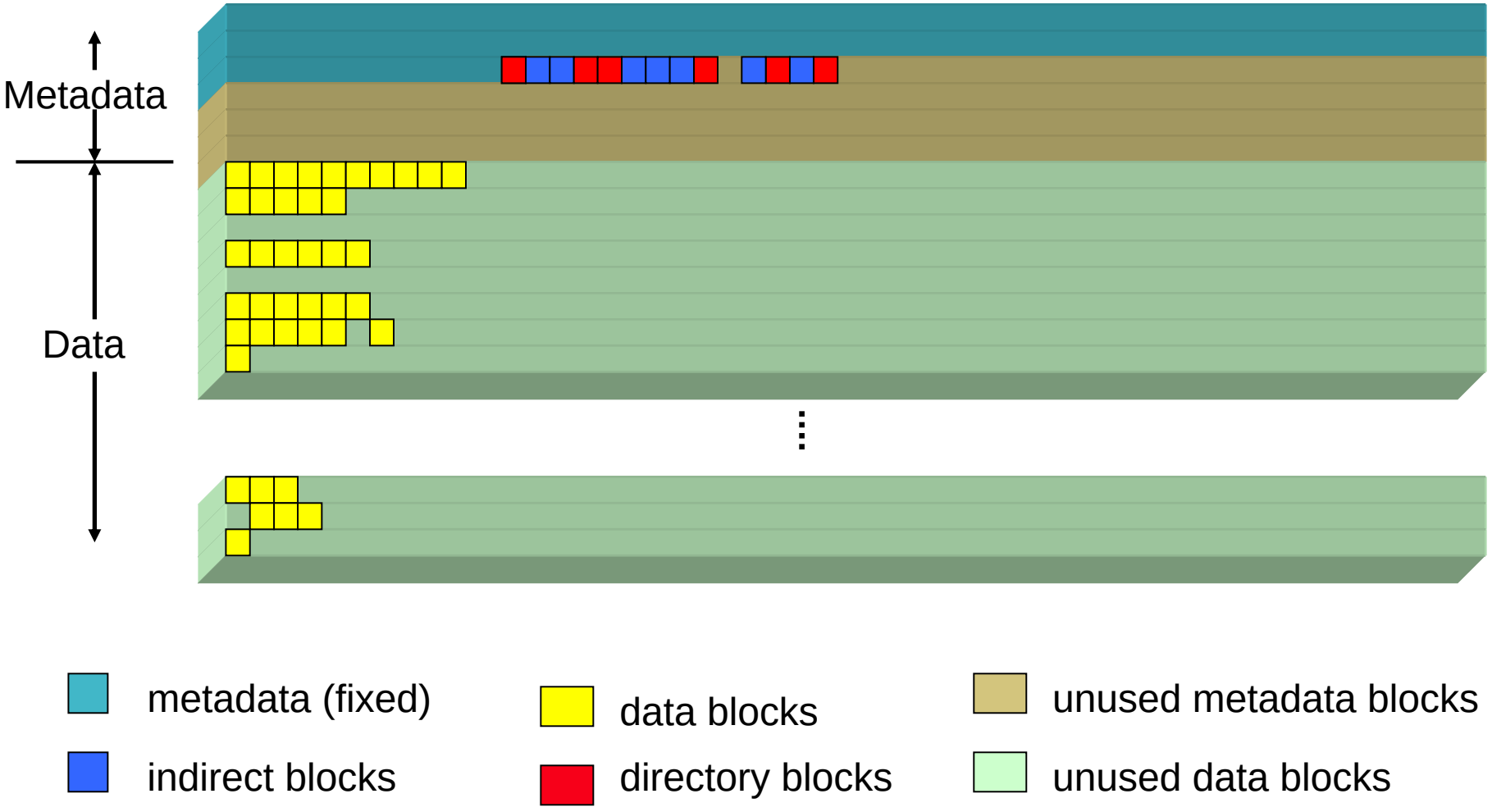
Why Separate Data and Metadata?

- Support different SLOs for data and metadata
  - Meta-data on fast storage for metadata intensive workloads
  - Encrypt file data without affecting metadata
  - Apply single-instancing to file data without affecting metadata
- Upgrade, modify, or reformat metadata more easily
- Support very large contiguous allocation for files
  - Improve sequential allocation of files
  - Consume existing data into file system as a file
- Support different block sizes for data and metadata
  - Small data block size (1k) for email apps
  - Large data block size (64k) for streaming apps
  - Metadata block size need not change

Block-Device File System (UbFS) Layout on Disk



Block-Device File System (UbFS) Layout on Disk (cont.)



Experimental Goals – Differing Security SLO

- Demonstrate the advantages of achieving differing service levels for data and meta-data for performance
- Provide superior security by full-disk data encryption with little or no metadata performance penalty
- Compare metadata operation performance of UxFS and UbFS
- Evaluate performance impact of encryption on file system operations while preserving SLO
- Demonstrate higher performance for metadata intensive workloads

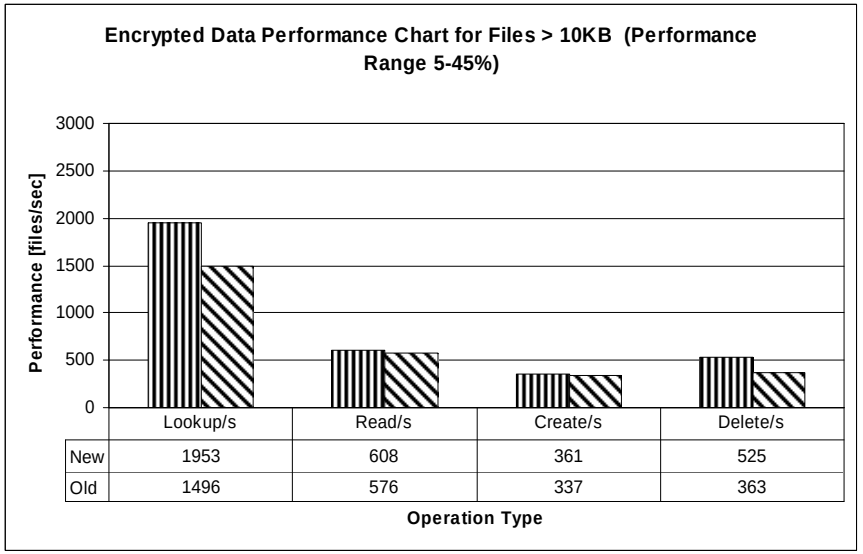
Experimental Goals – Differing QoS for Data and Metadata

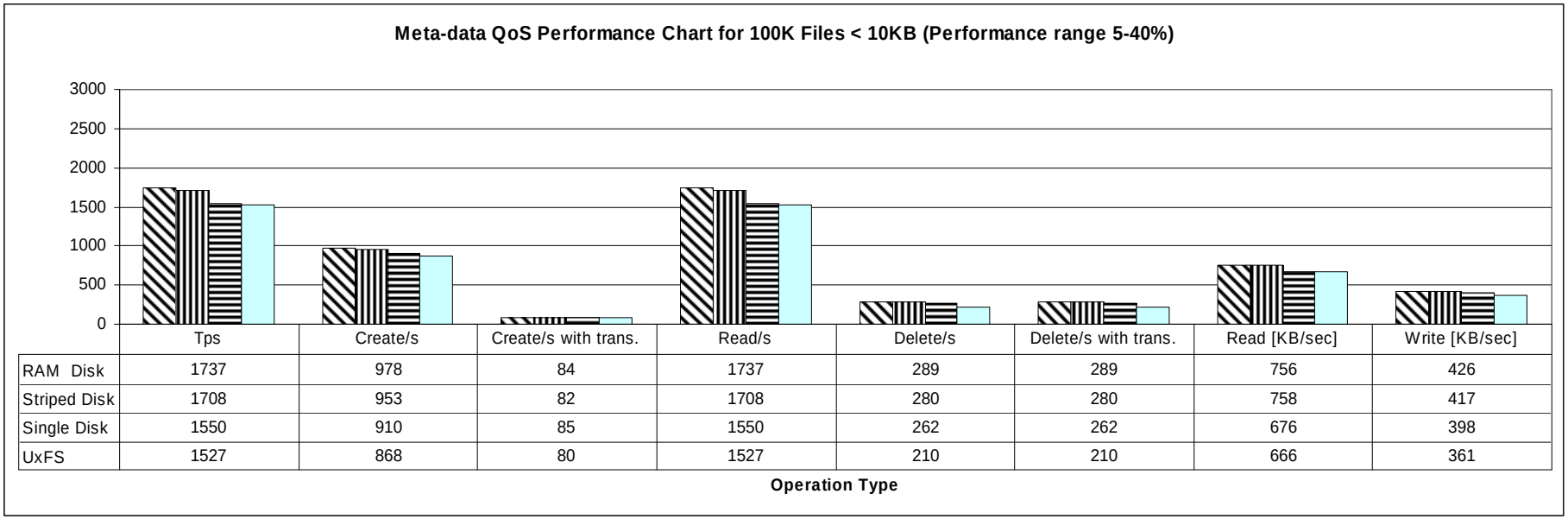
- Demonstrate the control on QoS for data by separation from metadata
- Characterize the performance of different metadata workloads for different classes of storage
- Characterize the performance of different classes of storage for data
- Compare the different QoS performance and reliability
- Evaluate performance of such tasks as fsck

Experimental Results

- Use prototype UbFS file systems with split data/metadata
- Concurrent dynamic support for both UbFS and UxFS file systems.
- Encapsulate existing LUN/volume structures
- Add imported files into a Version Sets
- Demonstrated higher performance using disk encryption
  - Data-only encryption
  - Metadata clear
- Demonstrate higher performance for metadata workloads
  - Metadata on high performance disk (RAID0)
  - Data on lower performance disk (RAID5)

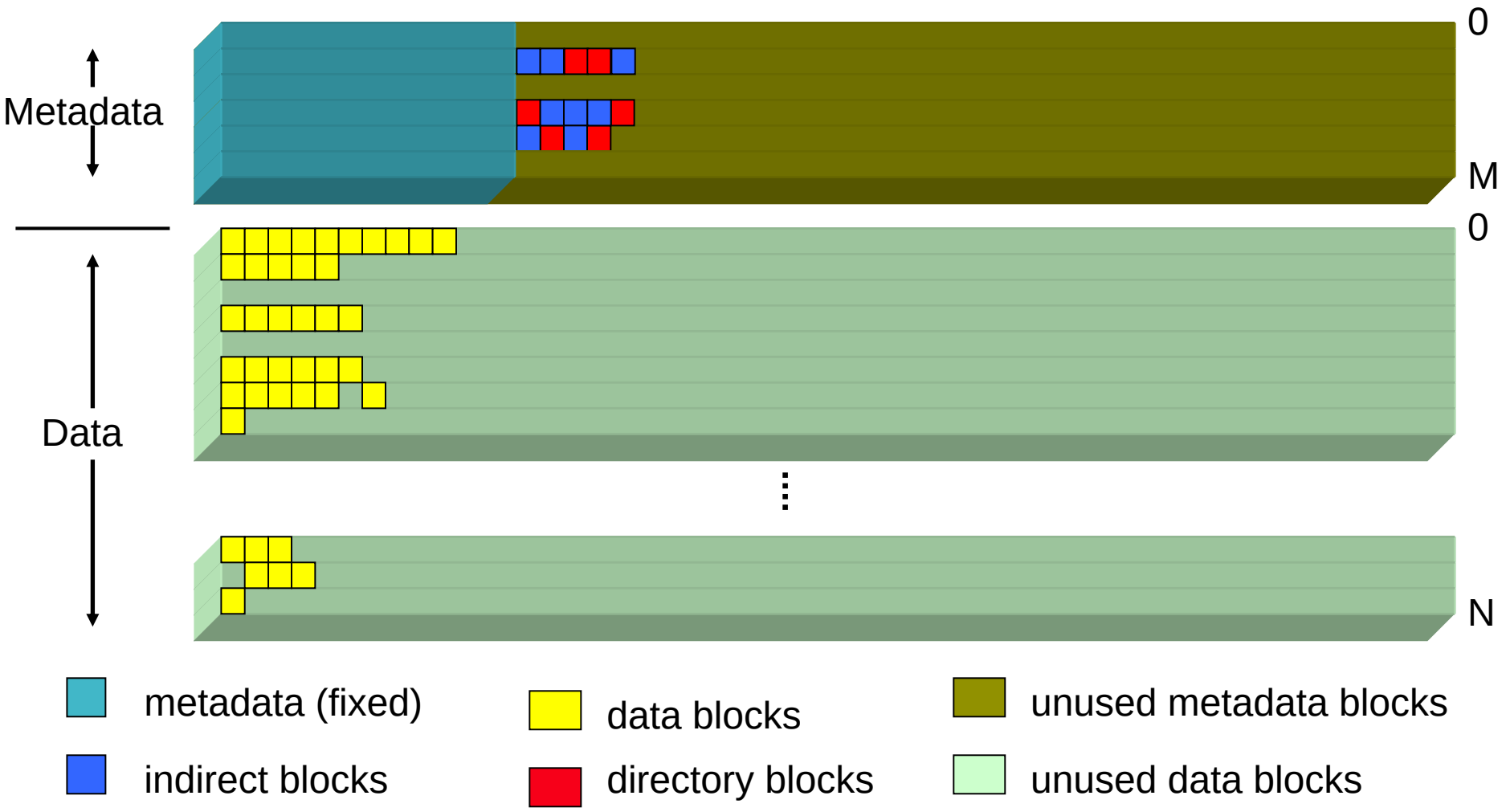
Experimental Results





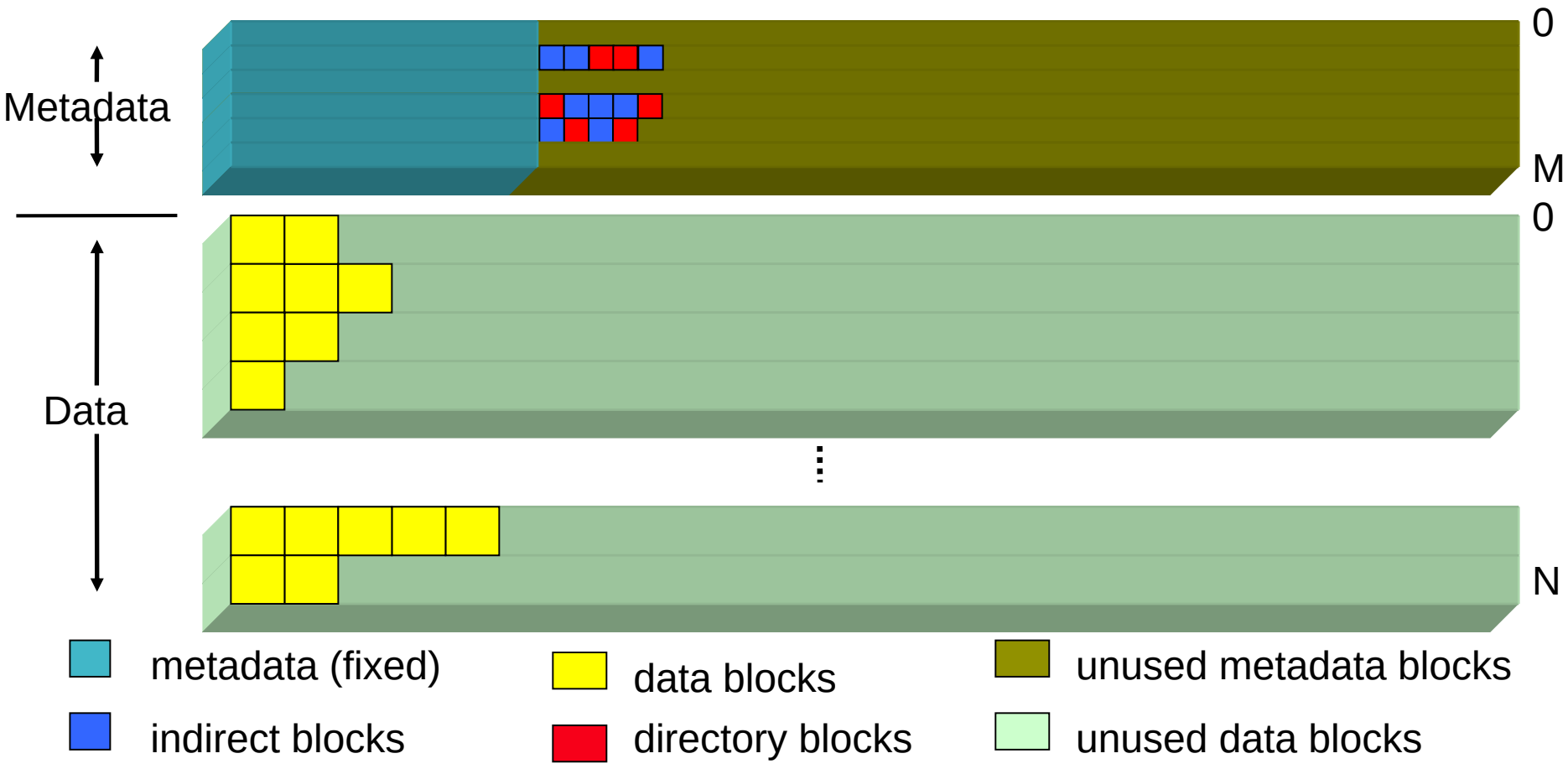
	Tps	Create/s	Create/s with trans.	Read/s	Delete/s	Delete/s with trans.	Read [KB/sec]	Write [KB/sec]
RAM Disk	1737	978	84	1737	289	289	756	426
Striped Disk	1708	953	82	1708	280	280	758	417
Single Disk	1550	910	85	1550	262	262	676	398
UxFS	1527	868	80	1527	210	210	666	361

Additional Features --- File System Extension



Additional Features --- Configurable Block Sizes

- Metadata block size different from data block size



Did we invent the wheel? – Related Work

- Dual FS is the closest to our approach applied to ext3
  - Linux FS
  - metadata organized as a log, separate from data volume
- QFS and ZFS (SUN) is using similar separation: “Metadata is stored on a separate device, reducing device head movement and rotational latency and providing the ability to mirror metadata only instead of all the data.”
- Cray was looking at a similar file system architecture which was implemented for the Fujitsu VPP5000 in 1998.
- Storage Tank uses mostly separation of the namespace and some file metadata but cannot be used on local disks
- PanFS mainly separate the dynamic metadata and not static and indirect blocks and indexing.


EMC²
where information lives®