ReFS

Lukáš Gemela

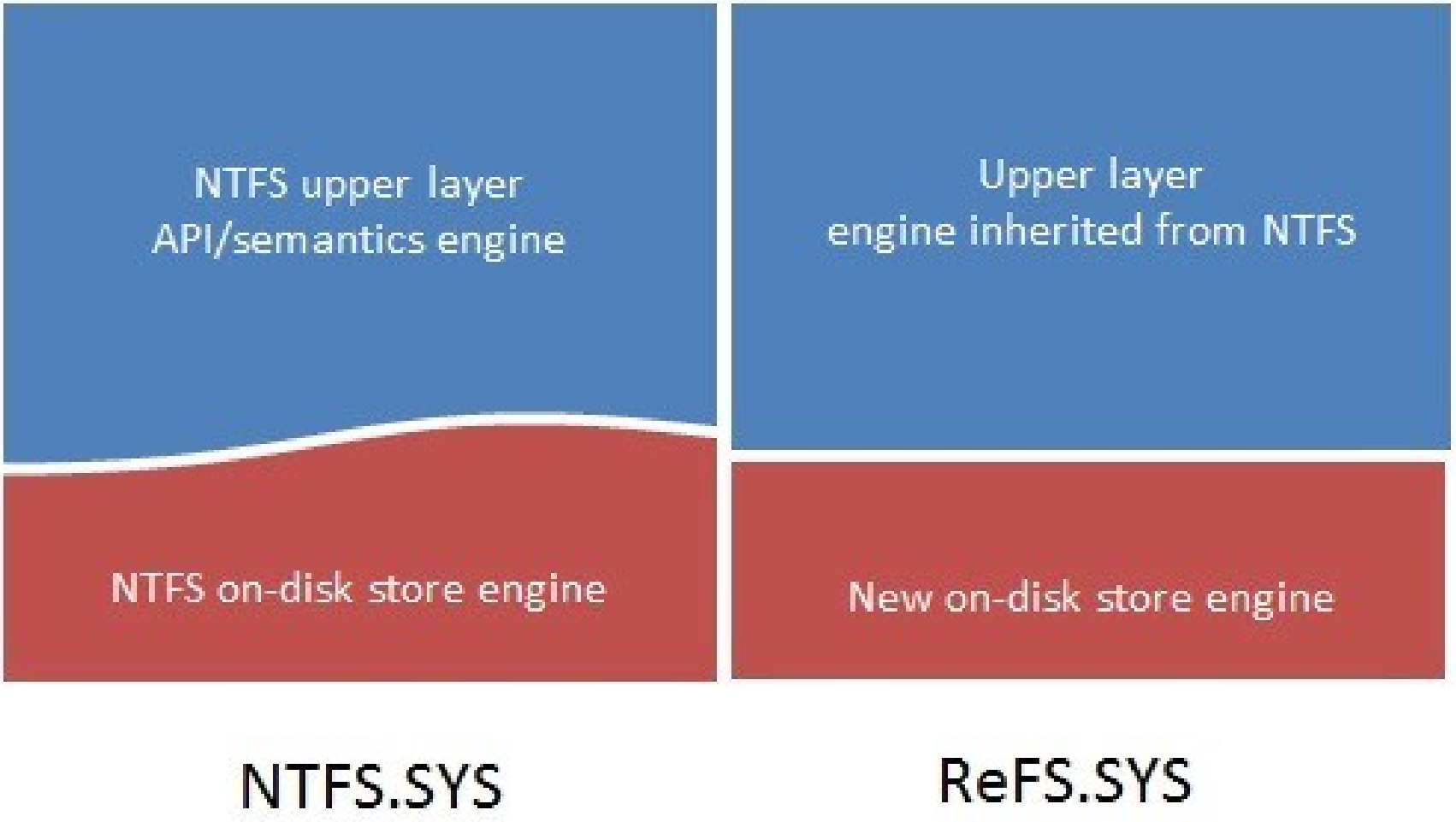
ReFS foundations

- ReFS = Resilient File System
- Nextgen file system for Windows
- Introduced in Windows Server 8
- Based on NTFS foundations
- Currently usable for file servers

ReFS key goals

- Compatibility with NTFS
- Verify and autocorrect data
- Optimize for extreme scale
- Never take the file system offline
- Storage-spaces feature

NTFS vs ReFS


NTFS upper layer
API/semantics engine
NTFS on-disk store engine
NTFS.SYS
Upper layer
engine inherited from NTFS
New on-disk store engine
ReFS.SYS

Interited features

- BitLocker encryption
- ACL
- USN journal
- Change notifications
- Oplocks
- Volume snapshots
- File ID
- Symlinks
- Mount points
- Same basic API as NTFS

Removed features

- Secondary streams
- Hardlinks
- Short names
- Compression
- ObjectID
- Sparse files
- Disk quotas
- Extended attributes
- EFS

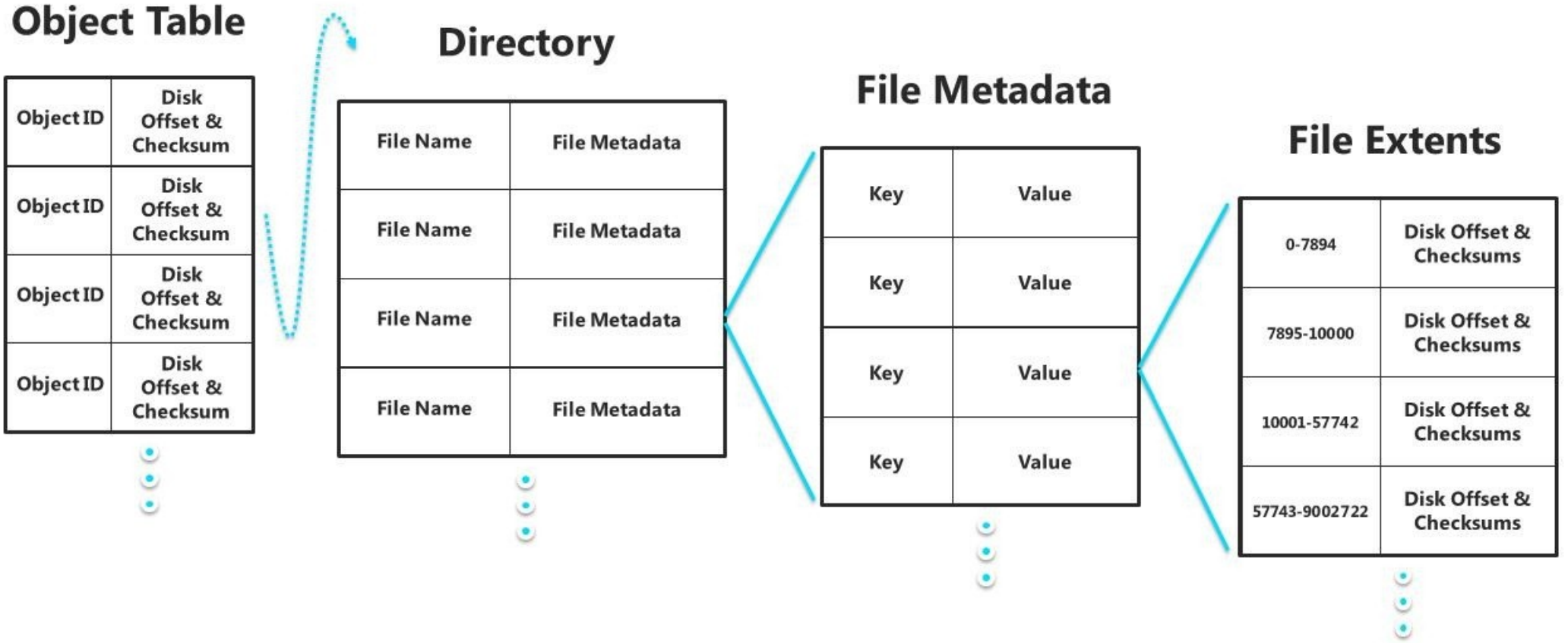
ReFS structure

- Generic Key-Value interface, notion „table“
- Implementation = B+ trees

- Benefits: scalable, simplifies the system, reduces the code

- Most tables have unique ID – Object ID

ReFS structure



Disk space allocation

- Hierarchical allocators
- Represents free space by table
- Table for small, medium, large memory chunks
- Accessible from Object table

Disk update strategy

- NTFS journal approach has some limits – e.g. „torn write“
- Allocate-on-write approach – never update metadata in-place!
- Write it to different location as atomic operation
- Transactions & journals are still present

Resiliency to corruptions

- All metadata is checksummed
- Can detect all forms of disk corruption, including lost and misdirected writes and bit rot
- Metadata checksum (64bit) always turned on
- Content of file can be checksummed as well → “Integrity streams“

Integrity streams

- Protect file content against all data corruption
- Checksums + copy on write approach
- Not appropriate for some cases (databases):

  some apps rely on a particular file layout

  some apps maintain their own checksums

  =>API to control the settings

Integrity streams API

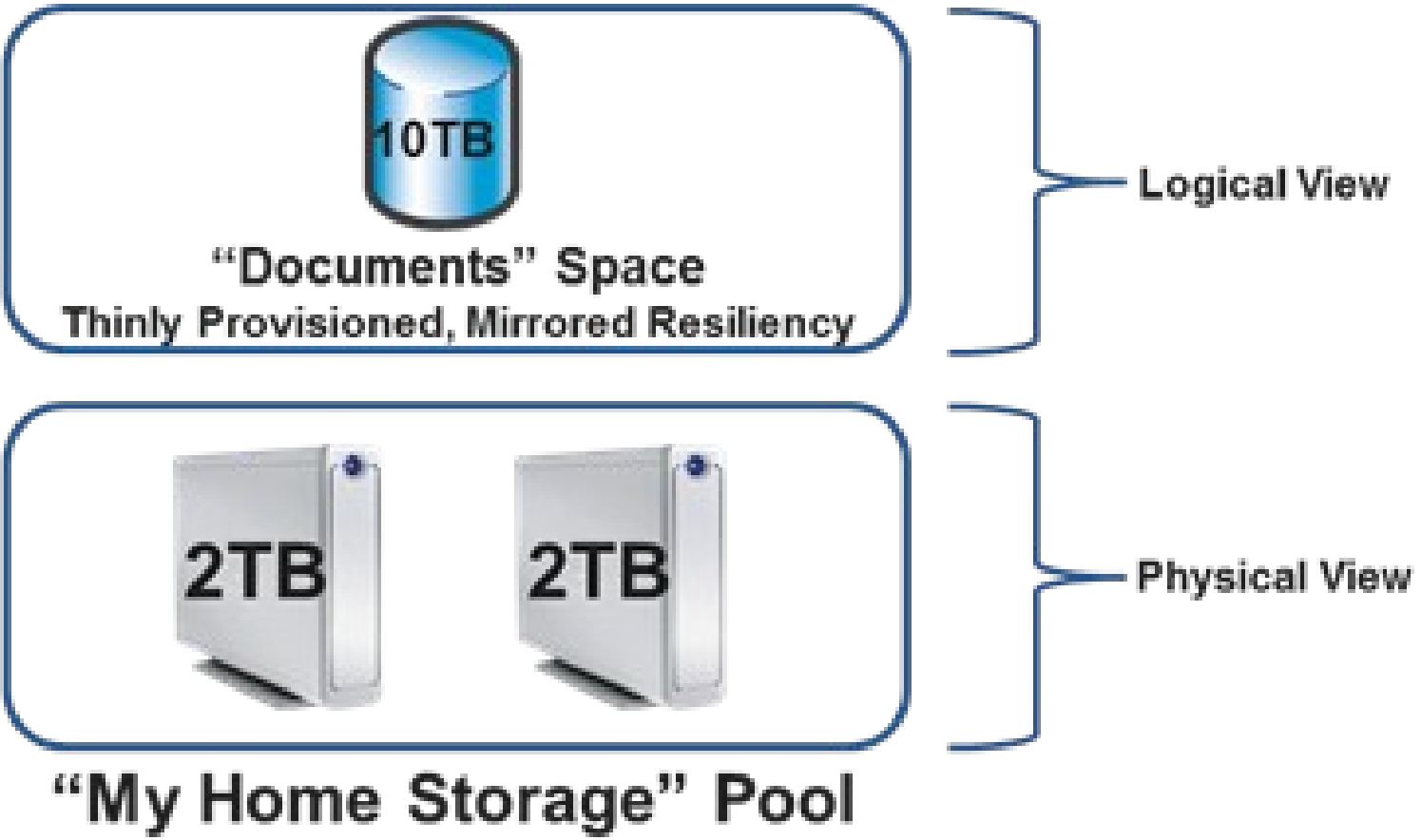
- FILE_ATTRIBUTE_INTEGRITY_STREAM
- Also attribute of a directory → inherited by all files inside the directory

  - D:\>format /fs:refs /q /i:enable <volume>
  - By default – depends if volume is mirrored

Storage spaces

- Allows to organize several physical devices into „Storage pool“
  - USB, SATA, Serial Attached SCSI
  - Storage pool can be composed of heterogeneous physical disks
  - Physical disks are no longer visible for Win
- Usage of virtual disks (spaces) from pool
- Resiliency through mirroring and parity

Storage spaces



Resiliency through mirroring

- Mirrored space
- We always store at least two complete copies on different physical disks within the pool
- Disk failure does not affect pool or Win at all
- Upon disk failure, data copies are regenerated for all affected spaces
- Hot-spare support

ReFS & Storage spaces

- They complement each other
  - ReFS detect such a failure (using checksums)
  - It interfaces with SP
  - SP reads all available copies and chooses the correct one (checksum validation)
  - SP fix all bad copies with the correct one
- ReFS without SP → data corruption event is logged

Battling bit rot

- „bit rot“ - data decay due to fact data was not read for a long time
- SP: system task, periodically scrubs all metadata and IS data on a ReFS volume
  - Involves reading and validating all the redundant copies
- FILE_ATTRIBUTE_NO_SCRUB_DATA

When all alse fails...

- ReFS implements „salvage“
  - removes all corrupted data
  - Non-repairable corruption does not affect the good-data availability
  - No Scandisk, no restarts, everything is online!
  - Can be completed in under a second

Limits

Maximum size of a single file	2^64-1 bytes
Maximum size of a single volume	2^78 bytes with 16KB cluster size
Maximum number of files in a directory	2^64
Maximum number of directories in a volume	2^64
Maximum file name length	255 unicode characters (compatible with NTFS)
Maximum path length	32K
Maximum size of any storage pool	4 PB
Maximum number of storage pools in a system	No limit
Maximum number of spaces in a storage pool	No limit

Usage and future

- Ready to be deployment-tested
- Not a beta feature
- Next step → storage FS for clients
  - Then as a boot volume
- In current stage it´s not possible to read ReFS volumes from W8 client or earlier (officially)
  - There are some „ways“

ReFS Disadvantages

- Not-bootable (yet)
- Not-usable for removable devices (yet)
- No direct data conversion from NTFS to ReFS
- Not fully compatible with NTFS
- Does not fit for databases or programs which are using special NTFS features (e.g. Steam)
- No support for UNIX

Sources

- http://blogs.msdn.com/b/b8/archive/2012/01/16/building-the-next-generation-file-system-for-windows-refs.aspx
- http://blogs.msdn.com/b/b8/archive/2012/01/05/virtualizing-storage-for-scale-resiliency-and-efficiency.aspx