Spectrum Scale: Metadata secrets and sizing revealed

IBM

Indulis Bernsteins

Systems Architect

indulisb @ uk.ibm.com

Madhav Ponamgi PhD

mzp @ us.ibm.com

2016

Spectrum Scale User Group

18 May *2016*

Notes

- Spectrum Scale Metadata often referred to as MD

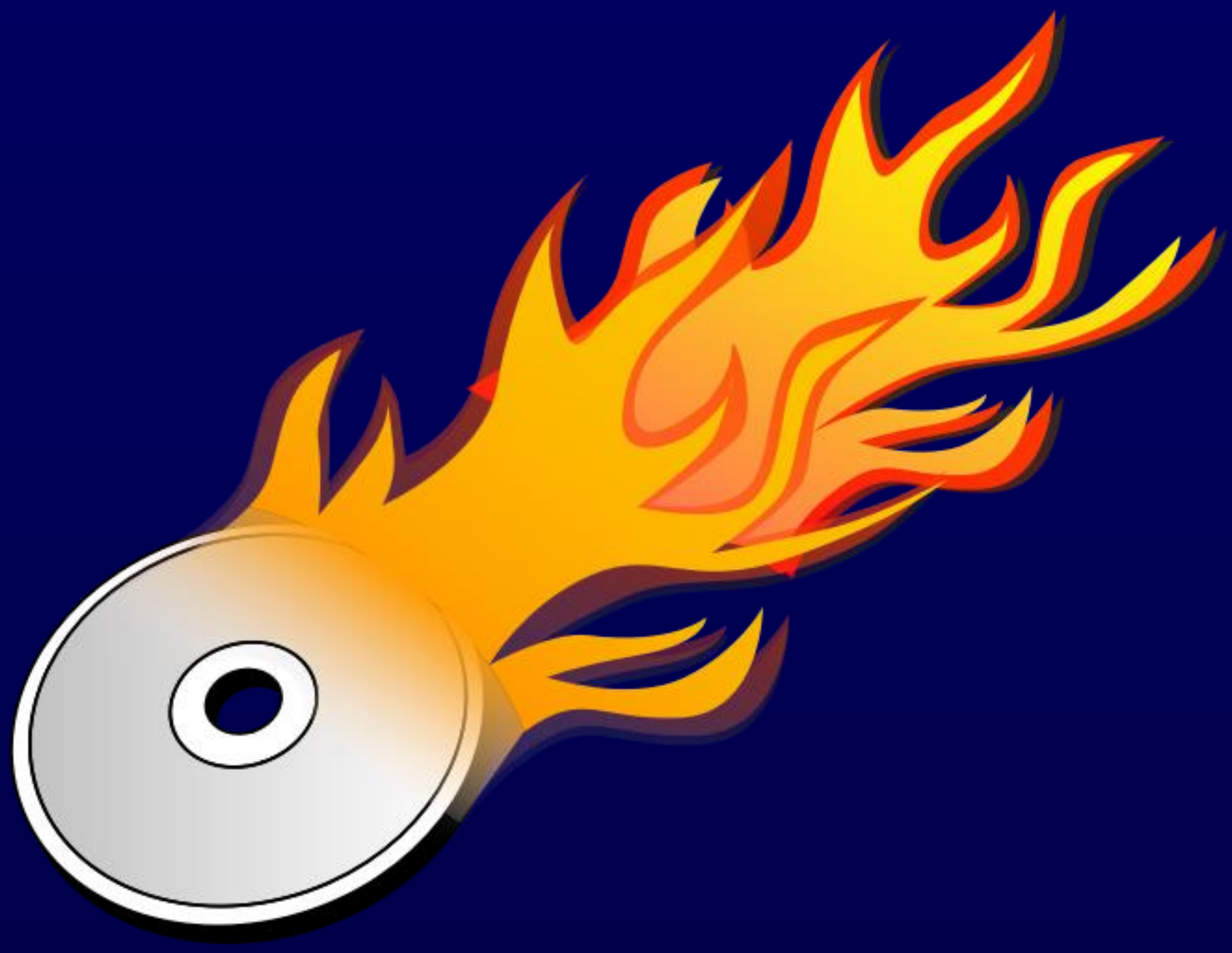
- Spectrum Scale can present the same single copy of Data as File, Object, Analytics data (HDFS) etc.
  - All Data is eventually stored in a File, ***File*** is used here to cover all types of Data stored

- Special thanks to Antoine Tabary, Ted Anderson, Yuri Volobuev, Madhav Ponamgi, Scott Fadden, Mark Roberts, Robert Garner, Eric Sperley, Lindsay Todd, Jess Jones

- Also see MD page on Developerworks http://ibm.biz/Bd4npd
  (some updates from this presentation have made it there already- *some*!)

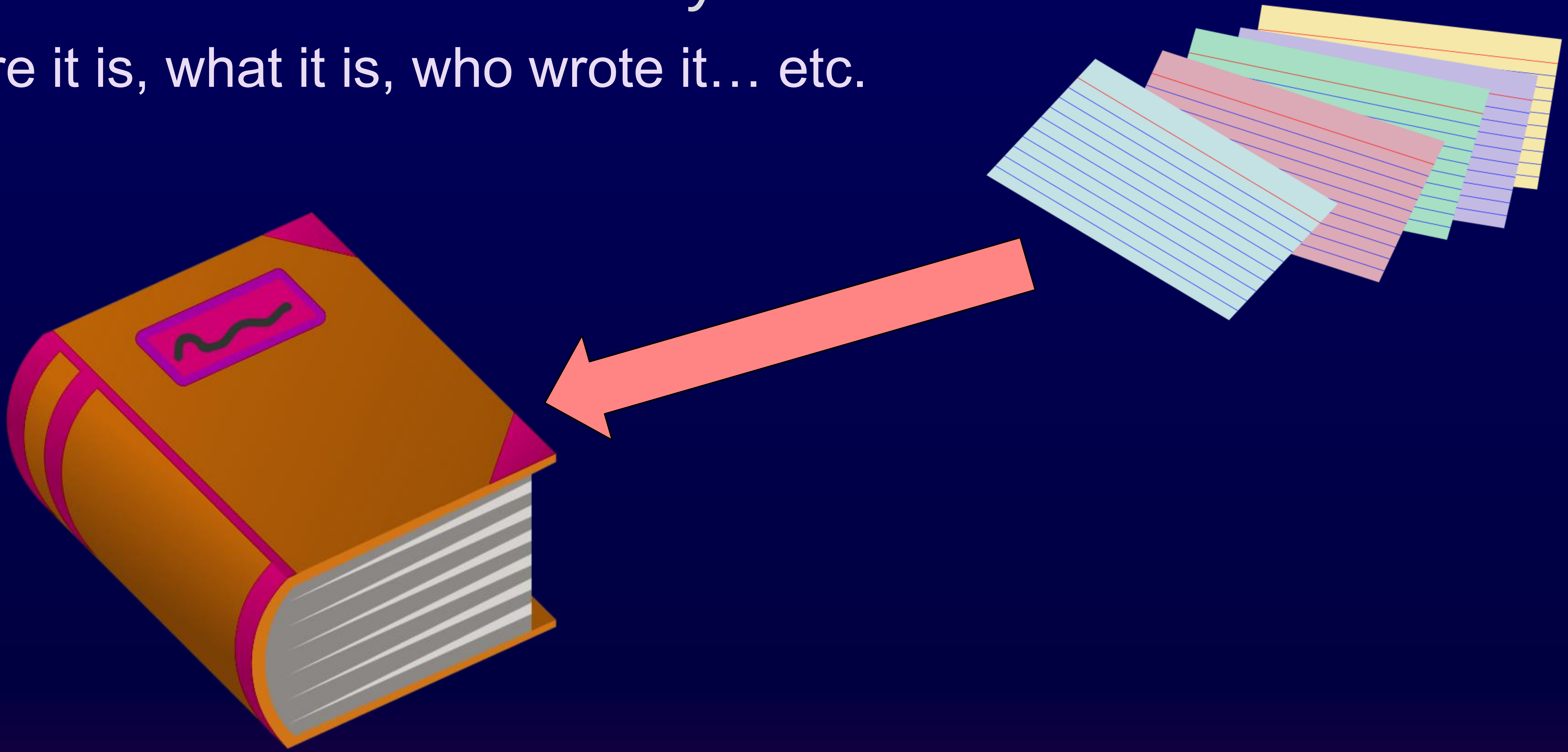
Burning Metadata questions

- What is it?
- How big should it be?
- How fast should it be?
- How do I do it “the best way”?



What is Metadata?

- Information associated with data, that is not the actual data
  - Describes the data in some way
    - Where it is, what it is, who wrote it… etc.



“Metadata” means different things to different people

- Filesystem metadata
  - *Directory structure, access permissions, creation time, owner ID, etc.*
- Scientist’s metadata
  - *EPIC persistent identifier, Grant ID, data description, data source, publication ID, etc.*
- Medical patient’s metadata
  - *National Health ID, Scan location, Scan technician, etc.*
- Object metadata
  - *MD5 checksum, Account, Container, etc.*

Filesystem Metadata (MD)

- Used to find, access, and manage data (in files)
  - Hierarchical directory structure
  - POSIX standard for information in the filesystem metadata (Linux, UNIX)
    - POSIX specifies what not how
      - Filesystem handles how it stores and works with its Metadata
    - Can add other information or functions, as long as the POSIX functions work

Why focus on Filesystem Metadata (MD)?

- Can become a **performance bottleneck**
  - Examples:
    - Scan for files changed since last backup
    - Delete files owned by user *courtney* **(**who just left the company)
    - Migrate least used files from the SSD tier to the disk tier
    - Delete snapshot #5, out of a total of 10 snapshots

Why focus on Filesystem Metadata (MD)?

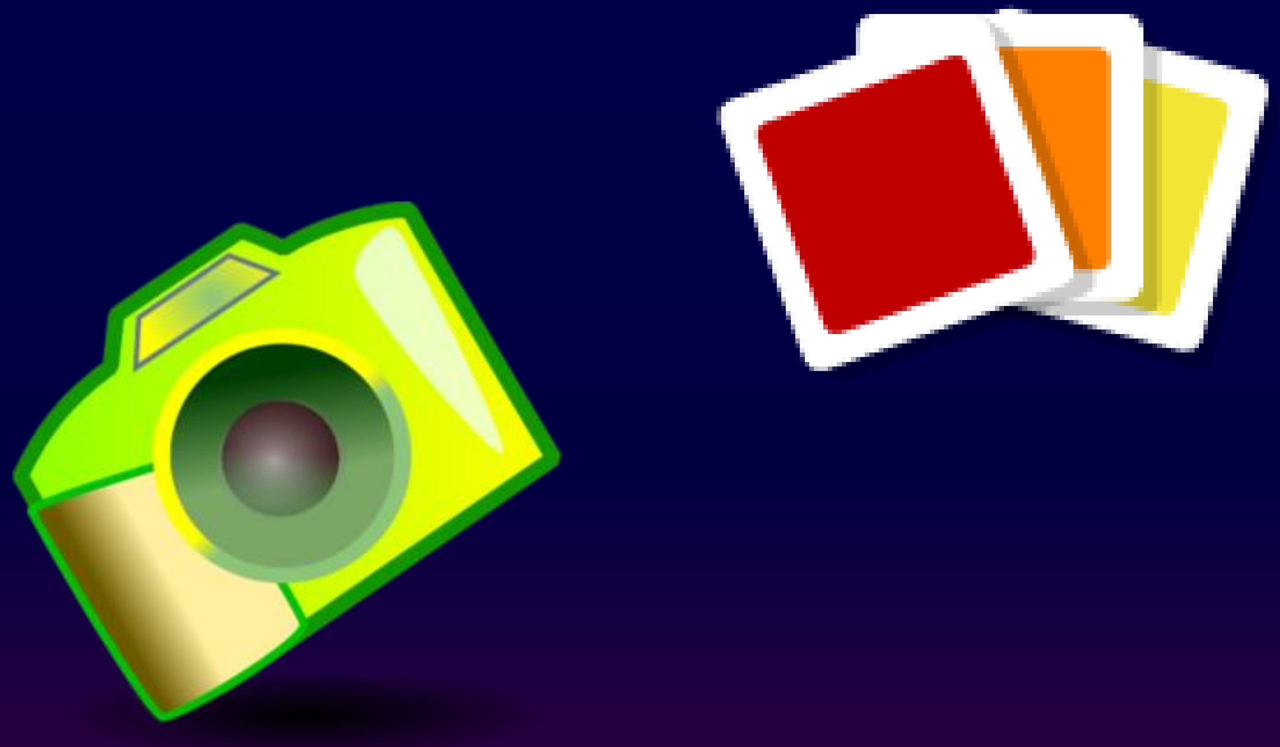
- Can be a **significant cost**
  - For performance, MD may need to be on Flash or SSD
    - Let’s try to get the **capacity** and **performance** right!

Spectrum Scale Filesystem Metadata (MD)

- POSIX compatible, including locking
- Designed to support **extra Spectrum Scale functions**:
  - Small amounts of file **data inside Metadata inode**
  - **Multi-site "stretch cluster**"
    - Via replication of Metadata and Data
  - **HSM / ILM / Tiering** of files to Object storage or Tape tier
    - Via MD Extended Attributes
  - **Fast directory scans** using the Policy Engine
    - Bypasses "normal" POSIX directory functions using MD directory structure
  - Other types of metadata using **Extended Attributes** (EAs)
    - EAs can "tag" the file with user defined information
  - **Snapshots**

Filesystem features which use additional MD capacity

- Extended Attributes (EAs)
- ILM/Tiering to offline storage pool (uses EAs)
- Data and MD replication by Spectrum Scale
  - Data replication: Max and “default” per filesystem
  - MD replication: Max and “default” per filesystem
- Snapshots



Other Spectrum Scale Metadata (besides filesystem MD)

- Filesystem Descriptor: stored on reserved area, multiple NSDs/disks
- Cluster Configuration Data: *mmsdrfs* file on server's local disk

  …etc

- We will only talk about **Filesystem Metadata**

Filesystem MD capacity

- Filesystem MD capacity is used up (mostly) by
  - Data: inodes = 1 per file

    + Indirect Blocks as needed (might take up a lot of capacity)
  - Directory information: inodes = 1 per directory

    + Directory Blocks as needed
  - Extended Attributes: in Data inode

    + EA blocks as needed

Extended Attributes (EAs): example use cases

- University of Pennsylvania
- Children’s Hospital of Philadelphia (CHOP)
- Radiology Departments
- Spectrum Scale (GPFS) system
  - 10Gig network
  - DCS 3700 / Native Raid Storage
  - Flash (mixed vendors)
  - Extended Attributes to classify and tag
    - Tumors
    - Fistula
    - Fissures

Penn
UNIVERSITY of PENNSYLVANIA

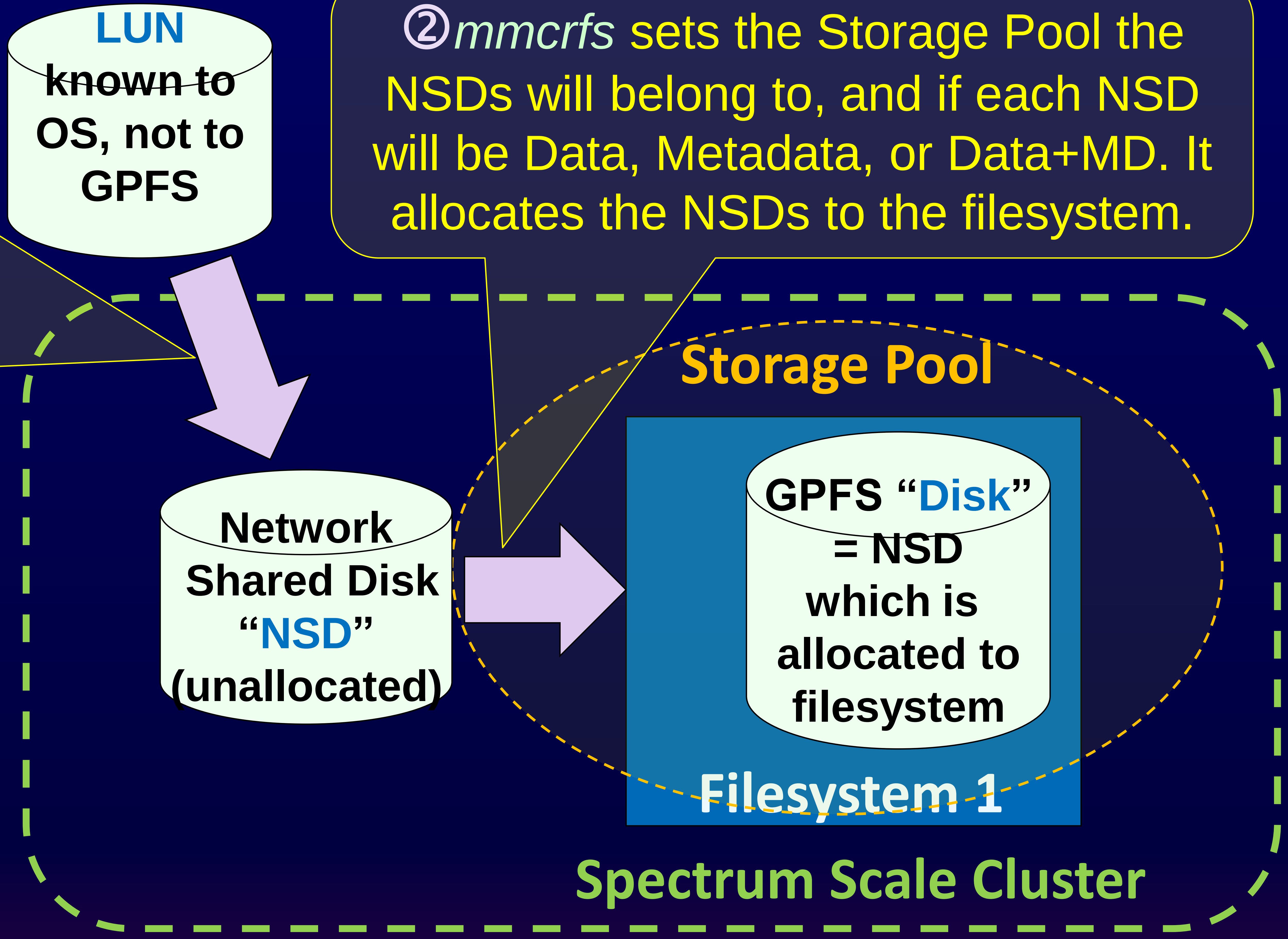
The Children's Hospital of Philadelphia®
Hope lives here.

NSDs and Storage pools

LUN ↔ "NSD" ↔ "Disk"

① *mmcrnsd* turns OS LUN ⇨ NSD, available to be allocated to a filesystem. NSD is not yet set to be Data-only, MD-only, or Data+MD. It is not yet allocated to a Storage pool.

② *mmcrfs* sets the Storage Pool the NSDs will belong to, and if each NSD will be Data, Metadata, or Data+MD. It allocates the NSDs to the filesystem.

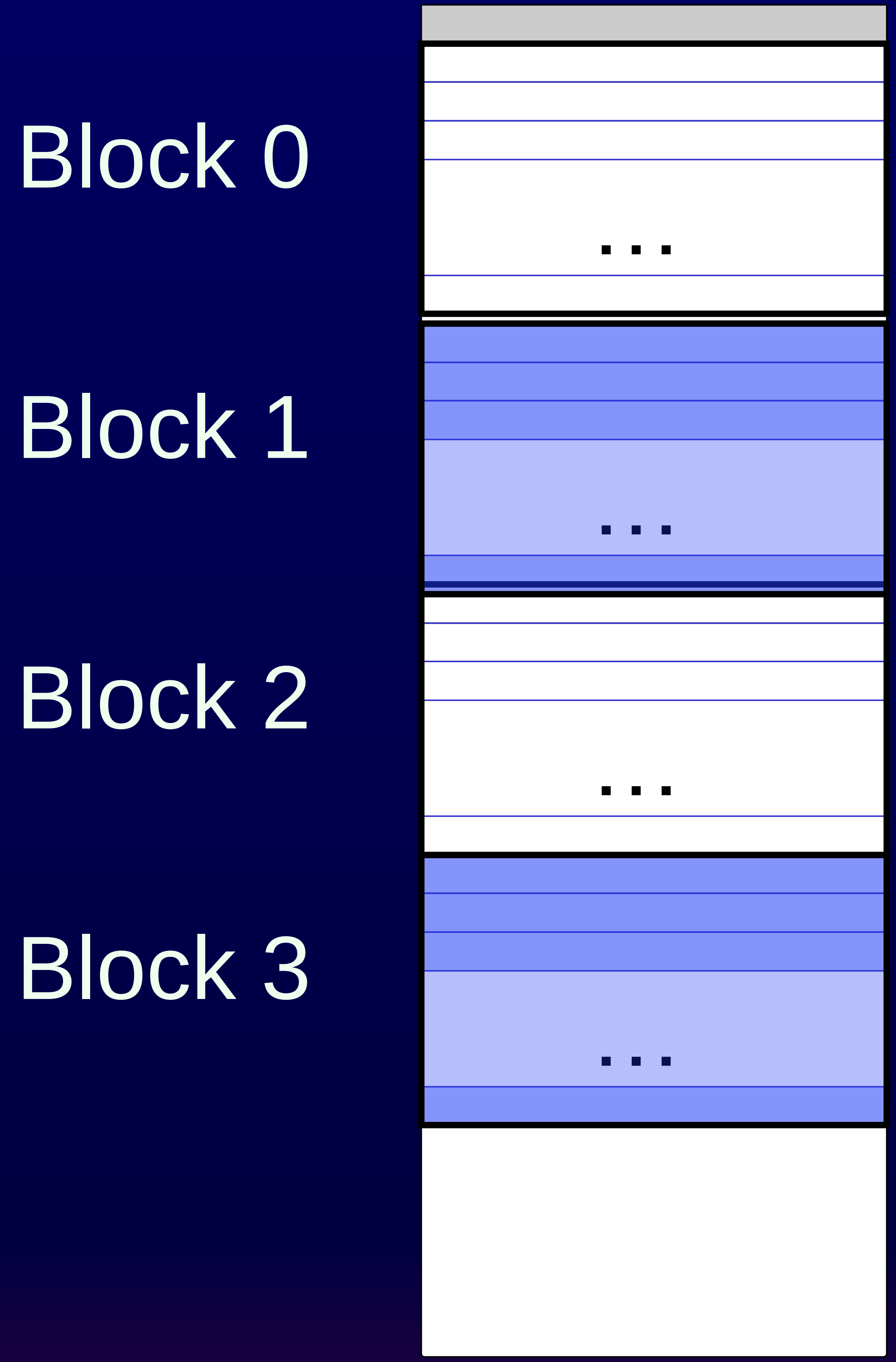


Data and MD space, on NSDs

- Operating System disks/LUNs allocated to Spectrum Scale become ***NSDs***
  - *Network Shared Disks* (even if they are not shared!)
  - Logically sliced into blocks and sub-blocks, based on MD or Data blocksize
    - Sub-blocks = $^{1}/_{32}$ of a block = “fragments”

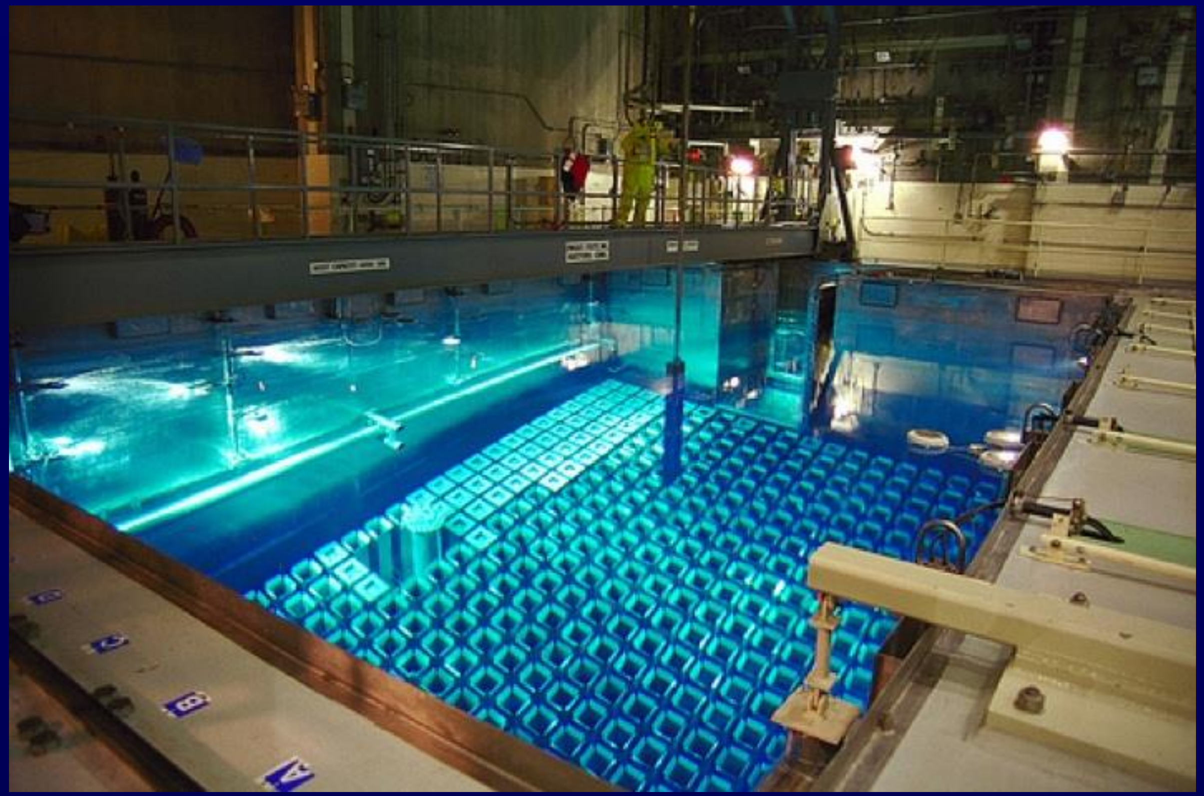
- NSDs allocated to a filesystem become Spectrum Scale “Disks”
    - Use *mm..disk* commands to manage, not *mm..nsd* commands
      - Migrate data or MD from one disk to another Disk etc.
      - Of course you can still list the NSDs *mmlsnsd*!

Basic NSD Layout: divided into fixed size blocks



- **Blocksizes** for Data and MD chosen at file system creation time
- Full block: largest contiguously allocated unit
- Subblock: $^1/_{32}$ of a block, smallest allocatable unit:
  - Data block fragments (one or more contiguous subblocks)
  - Indirect blocks, directory blocks, …
- Choices: 64k, 128k, 256k, 512k, 1M, 2M, 4M, 8M, 16M
  - ESS GNR vdisk track size must match filesystem blocksize
    - 3-way, 4-way replication: 256k, 512k, 1M, 2M
    - 8+2P, 8+3P: 512k, 1M, 2M, 4M, 8M, 16M

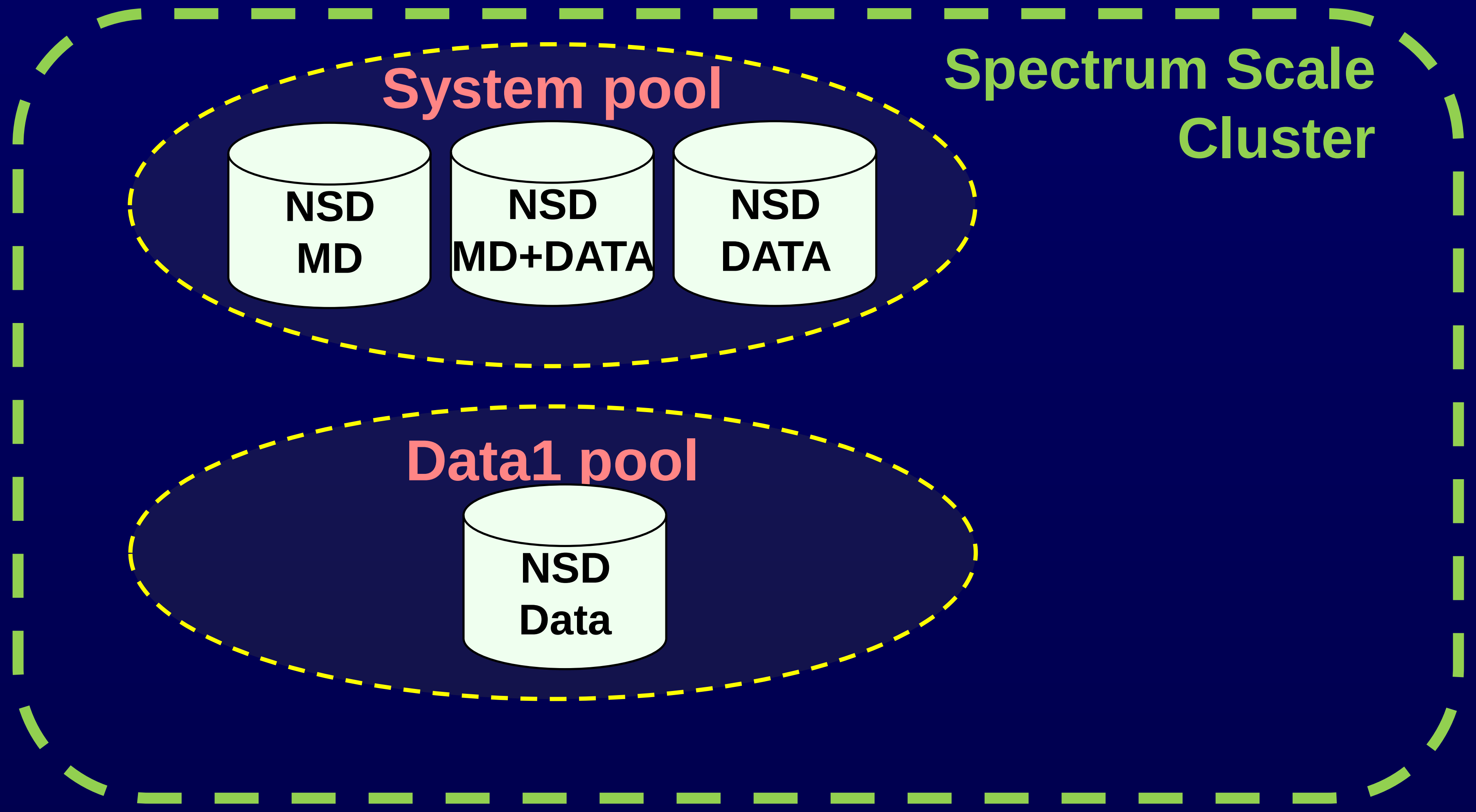
Spectrum Scale Storage pools



- System LUN ⇨ Spectrum Scale “NSD”
  ⇨ “Disk” when allocated to filesystem
  - An NSD belongs to a Storage Pool
- A Storage Pool can have many NSDs in it
- NSDs in a Storage Pool can be allocated to many different filesystems
  - But each NSD can only belong to a single filesystem

System and other Storage pools

- System storage pool, 1 per cluster
  - Only System pool can store MD
  - System pool can also store Data
    - NSDs defined as Metadata Only, or Data Only, or Data And Metadata
- User-defined Data storage pools, 0, 1 or many
  - Data only, no MD
  - All NSDs must be defined as Data Only
  - Filesystem default placement policy required



Where does Metadata (MD) live?

- MD is **only** stored in the System storage pool
  - One System pool per cluster, can contain multiple NSDs

- Each NSD/disk in the System pool is defined as **either**:
  - Data only
  - Metadata Only
    - Filesystem with no Data+MD NSDs, MD blocksize can be chosen separately from Data
    - Typically 64 KiB, 128 KiB, or 256 KiB
  - Data And Metadata
    - Filesystem using Data+MD NSDs, MD blocksize = filesystem (Data) blocksize
    - Note: Large fileystem blocksize e.g. 16 MiB often chosen for Data performance
      - Can lead to wasted space & excessive MD capacity usage, due to size of Indirect Blocks

Metadata (MD) creation

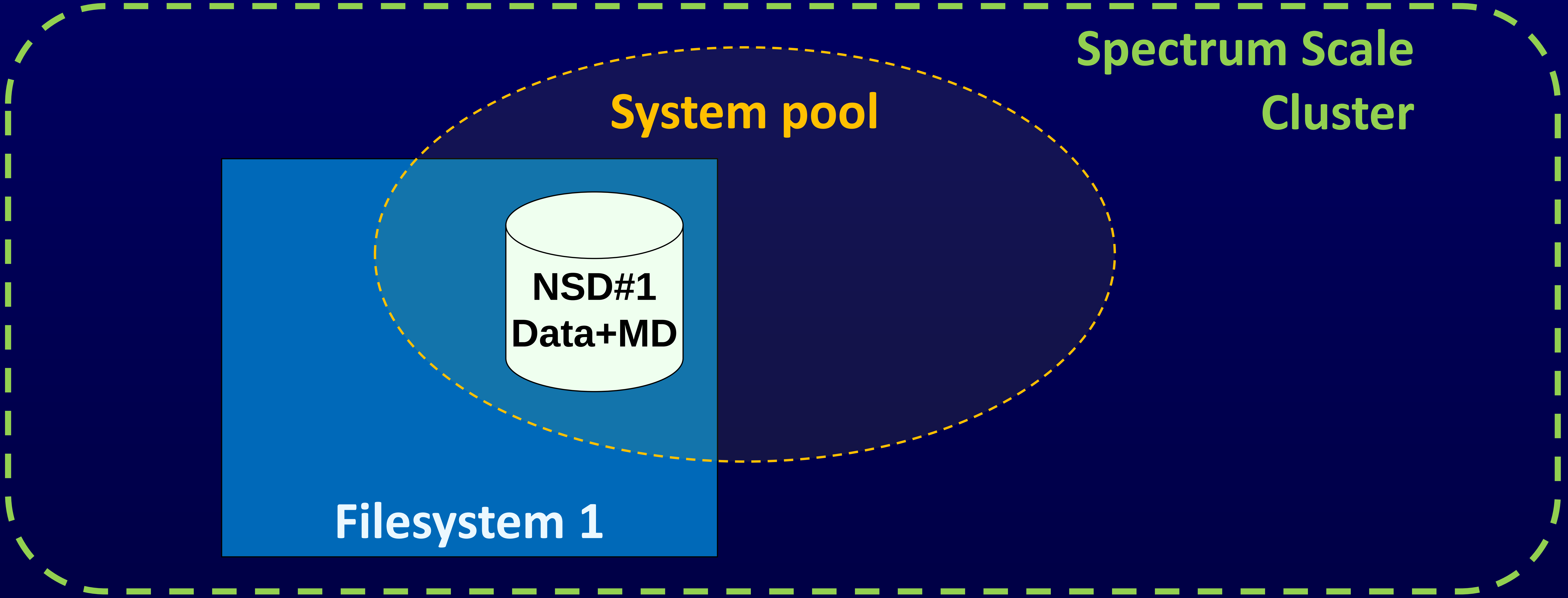
- Metadata characteristics are set at filesystem creation
  - Blocksize for MD
    - Optionally different to Data blocksize
  - inodes
    - inode size (512 bytes, 1K, 4K)- can't be changed later
    - inodes can be pre-allocated
  - **Not all MD space is used for inodes**
    - ***…so do not pre-allocate all of MD for inodes, leave room for Directory Blocks, Indirect Blocks etc***

Filesystem with separate MD and Data blocksizes

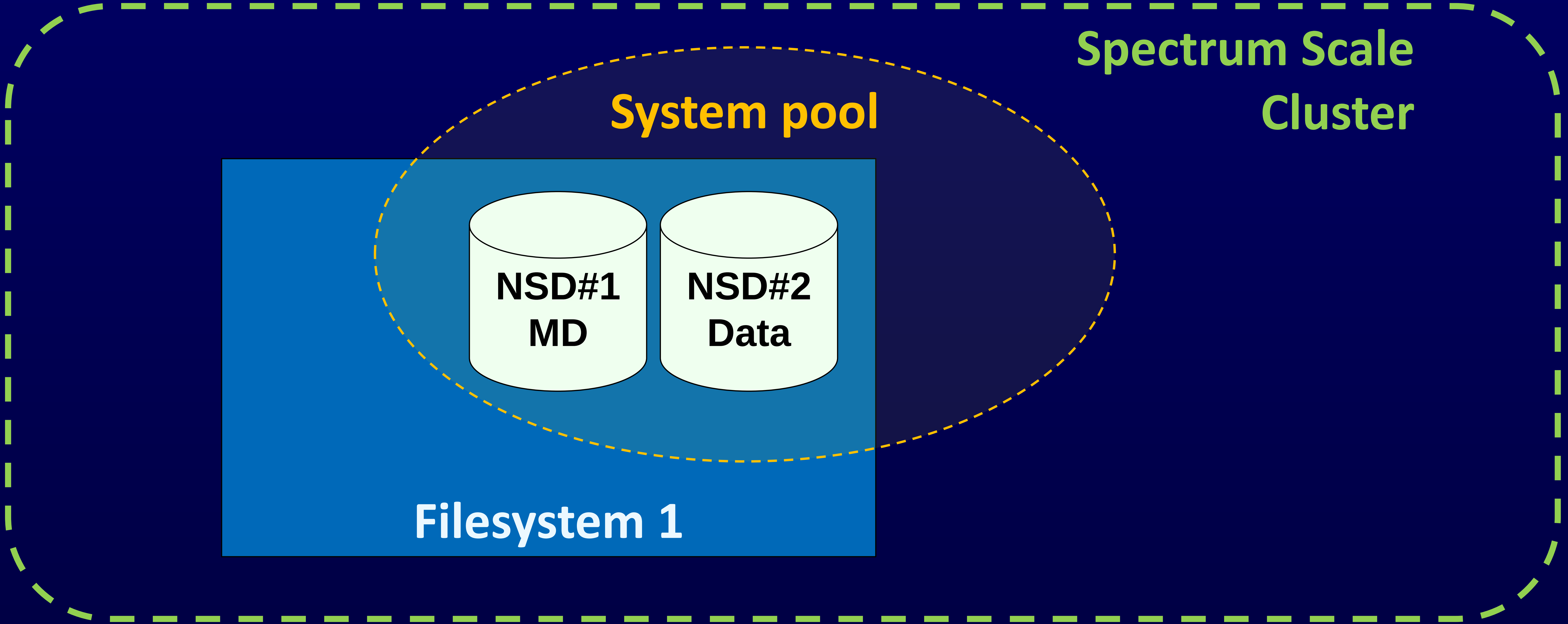
- Large blocksizes (e.g. 16 MiB) are often chosen for Data performance
  - If there are many large files, can lead to excessive MD capacity usage
  - Can lead to low performance- small MD writes to large RAID stripes

- To create a filesystem with an MD blocksize different from Data blocksize
  - Select **<u>only</u>** NSDs which are **MD-only** for Metadata
  - Use *mmcrfs … --metadata-block-size 256K* when creating the filesystem
  - Typical MD blocksizes are 64 KiB, 128 KiB, or 256 KiB (coul dbe up to 16 MiB)
  - If **<u>any</u>** Data-and-Metadata NSDs are selected for Metadata, cannot specify a separate MD blocksize, will be one blocksize for Data + MD

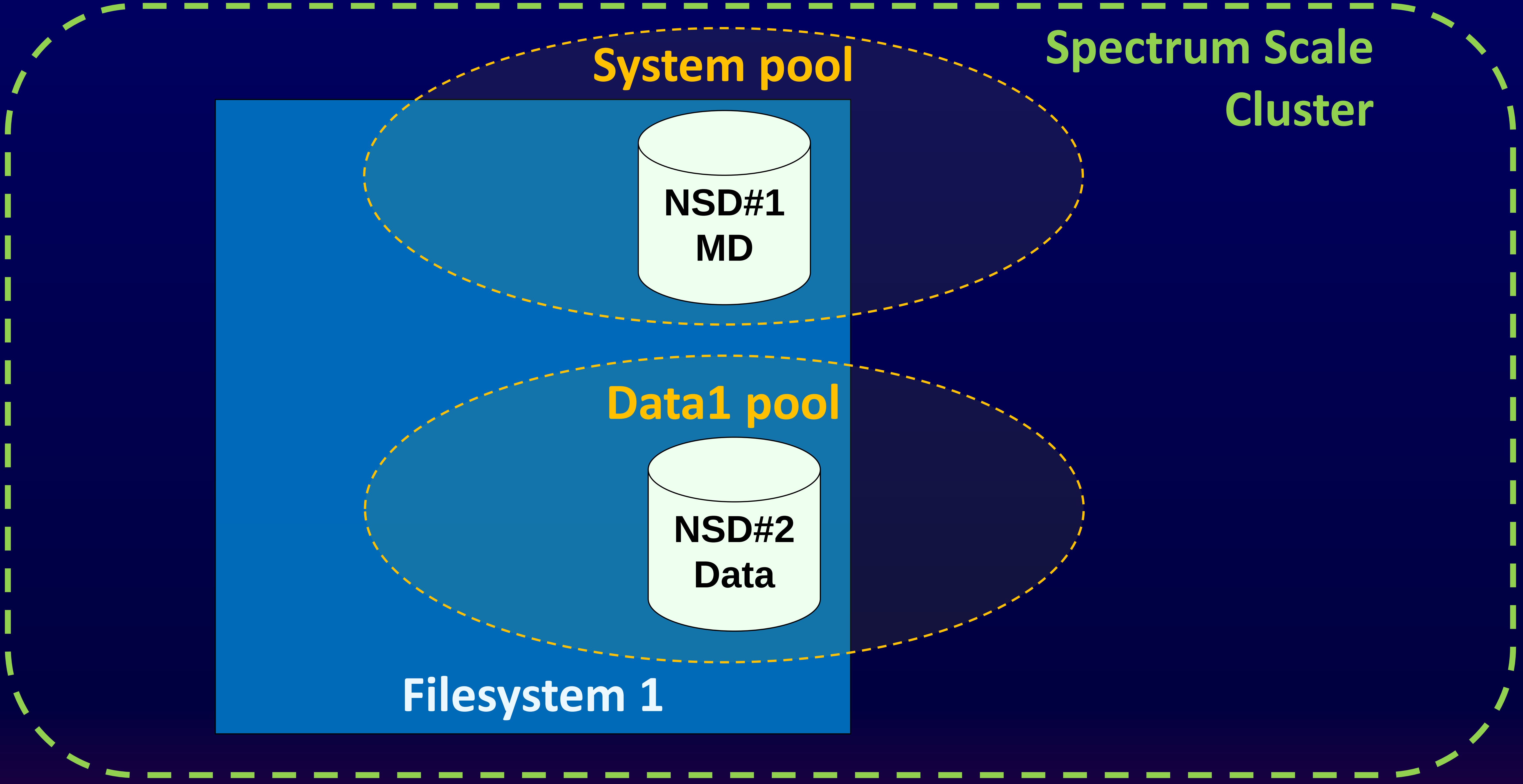
Where does Metadata live? Example #1



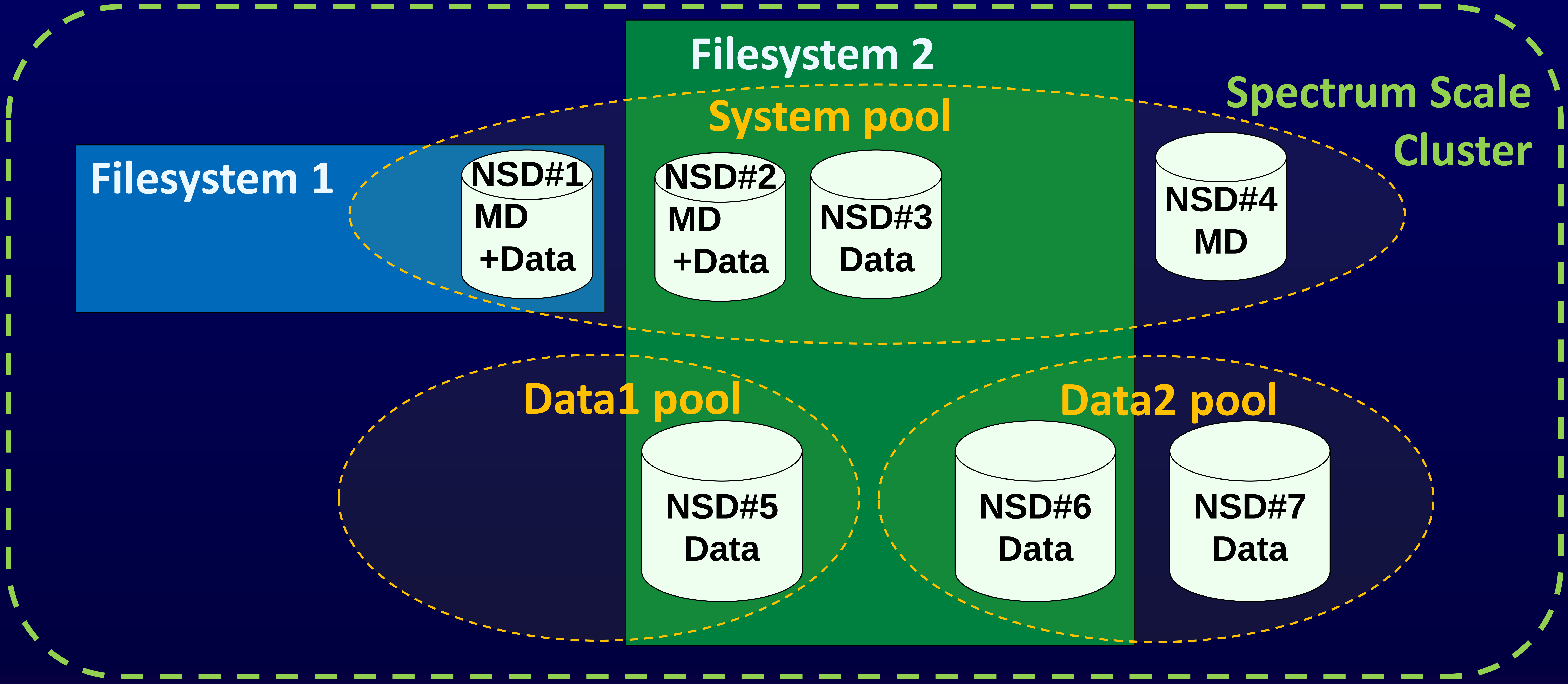
Where does Metadata live? Example #2



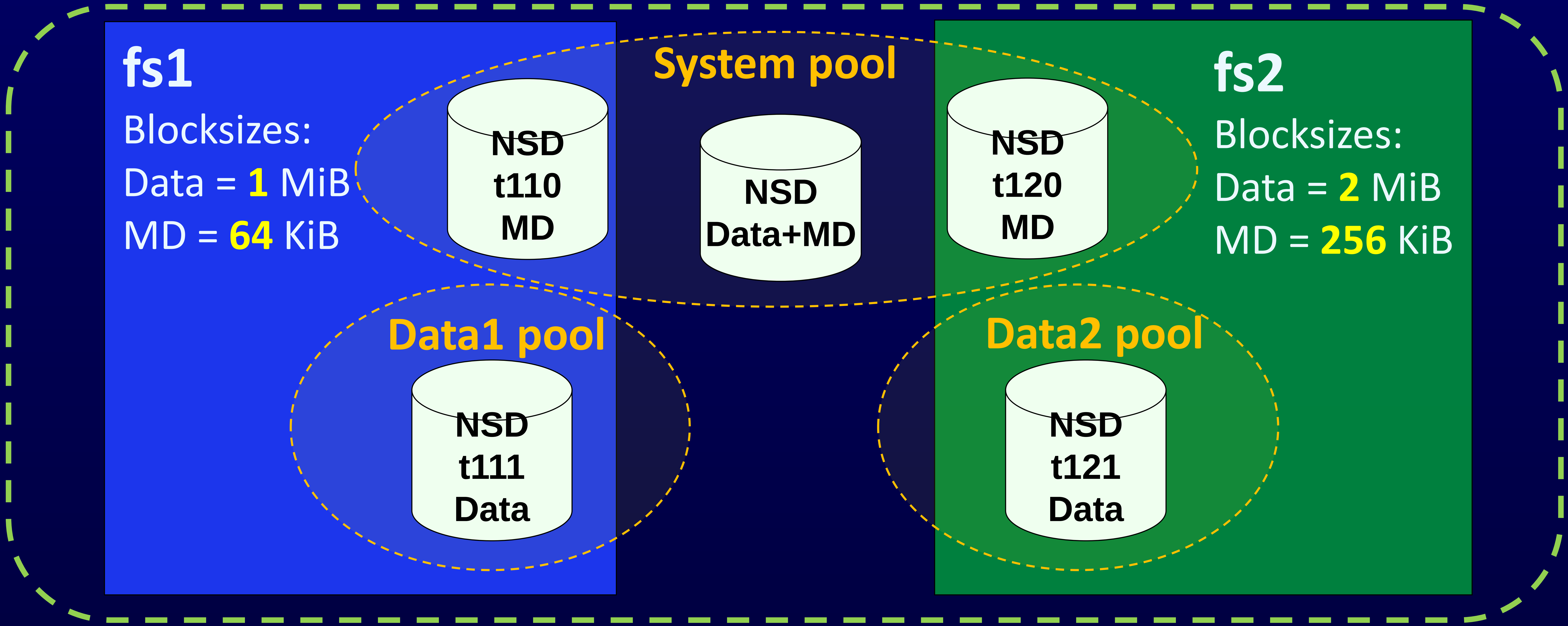
Where does Metadata live? Example #3



Where does Metadata live? Example #4



Different MD blocksizes: Example #5



2016: A Metadata Space Odyssey

My God… it's full of files!

What is in Filesystem Metadata space?

- Metadata capacity allocated as needed to:
  - inodes: for files and directories
    - inode = one block in the "invisible" inode file
  - Indirect blocks
  - Directory blocks
  - Extended Attribute block
  - etc.

All MD information is ultimately held in "files" in MD space. Some are fixed size, some are extendable, some are accessed using normal file access routines, others using special code

Other Metadata stuff- “high level files”

- Stored as files, not visible to users
  - Same locking as normal files
    - But uses special read/write code

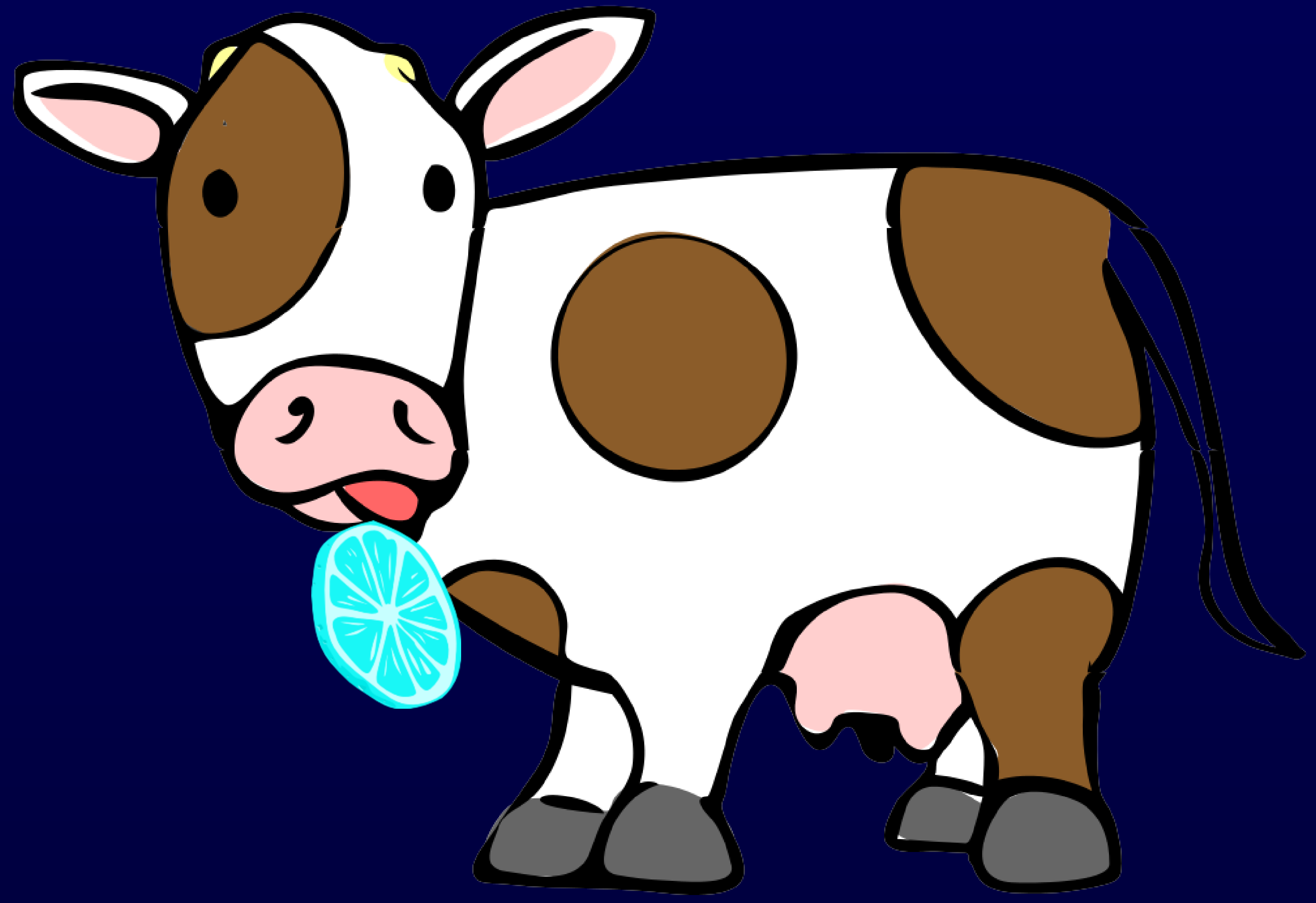
- inode file
- directory files
- ACL file
- Quota files
- Allocation summary file
- Extended Attribute file
  - GPFS 3.3 and earlier

Other Metadata stuff- “low level files”

- Files, not visible to users, special read/write and locking code
  - Log files
  - Fileset metadata file, policy file
  - Block allocation map
    - A bitmap of all the sub-blocks in the filesystem, 32 bits per block
    - Organised into n equal sized “regions” which contain some blocks from each NSD
      - This is what *mmcrfs -n NumNodes* does
      - Node “owns” a region and independently do striped allocation across all disks
      - Filesystem manager allocates a region to a node dynamically
  - inode allocation map
    - Similar to block allocation map, but keeps track of inodes in use

What is <u>not</u> in Metadata space?

- "Metadata space" does not include actual data
  - Unless it is very small, in which case it can hide inside the inode!
    - Just like a small Blue orange hides inside a cow…



I don't know why a Cow and a Blue Orange appear in a Metadata presentation. Office thinks the graphics are somehow related to Metadata!

inodes

- "The term *inode* is a contraction of the term *index node*..."
  - *Maurice J. Bach, The Design of the Unix Operating System, 1986*

- "the *inode* is a data *structure* used to represent a filesystem object, which can be one of various things including a file or a directory" https://en.wikipedia.org/wiki/Inode
  - POSIX filesystem architecture is based on inodes, flexibility in underlying structures and design
  - POSIX can be fully or partially emulated on other filesystems e.g. NFS export of NTFS

- "Main" information about a file or directory, and how to find related data
  - inodes for a file
    - Most POSIX filesystems do not generally hold data in an inode, or directory entries in an inode
      - Spectrum Scale can hold a small amount of data, or directory info in the inode

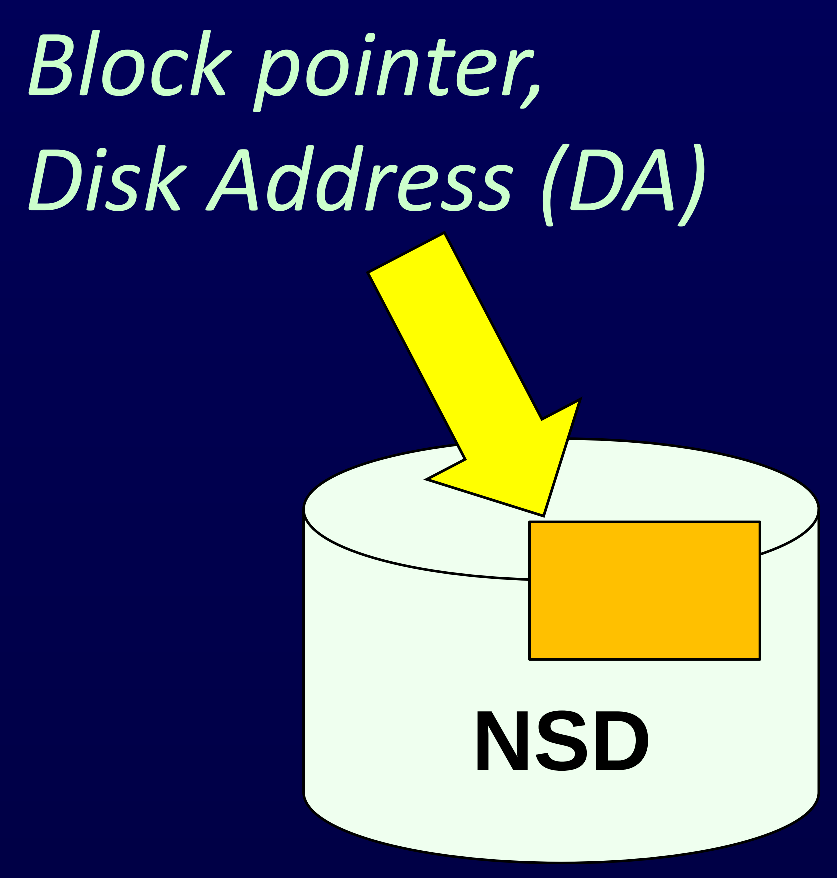
inodes

- Fixed size “blocks” which hold some (not all!) filesystem metadata
  - Allocated on System storage pool (NSDs)- metadataOnly or dataAndMetadata
  - 512, 1024, or 4096 Bytes each: set at filesystem creation, **cannot be changed later**
  - Held in one invisible inode file, extended as required

- inodes are used for Data or Directory use, and can contain:
  - Disk block pointers = Disk Addresses = DAs
    - ..or pointers to Indirect blocks which then point to Data blocks
  - File data (for very small files)
  - Directory entries, or pointers to Directory blocks
  - EAs, or pointers to EA blocks

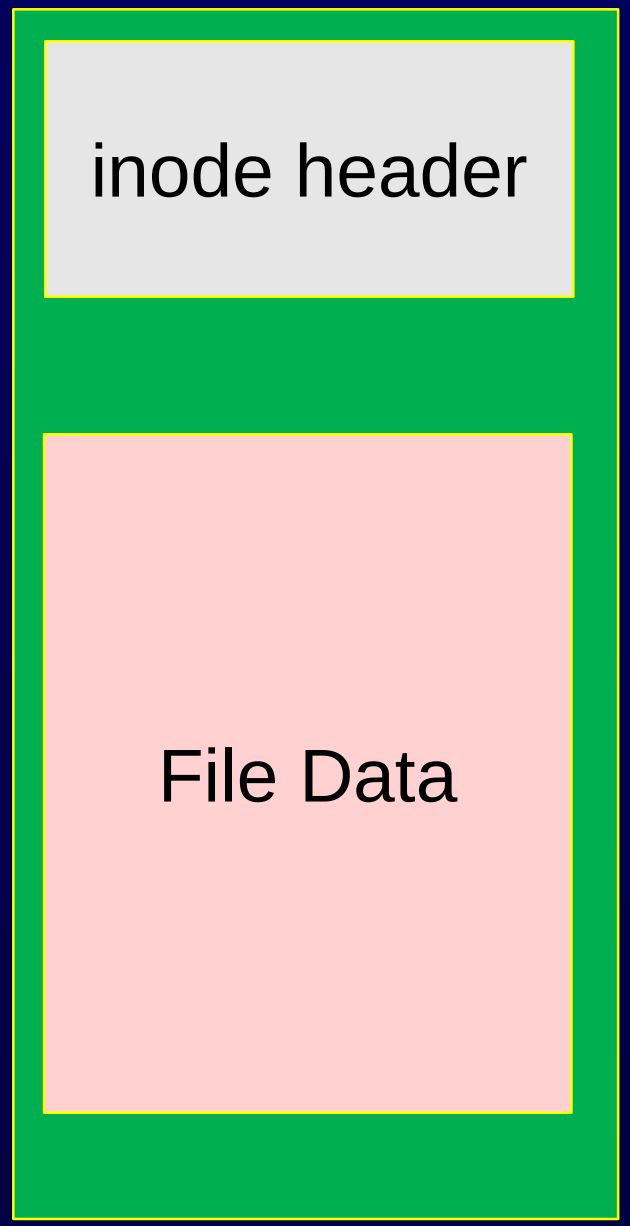
Finding the Data

- Find file data by following pointers to disk blocks or sub-blocks
  - Pointers to Data blocks on NSDs are Disk Addresses (DAs) = 12 bytes
    - Points to a block or sub-block on an NSD
- Very small files fit into spare space in an inode
  - No need for pointers to Data blocks
- Larger files
  - Store pointers/DAs to Data blocks/sub-blocks in an inode
  - Even larger… store pointers to Data in a block on an NSD = "Indirect Block"
  - Even larger… store pointers to Indirect Blocks

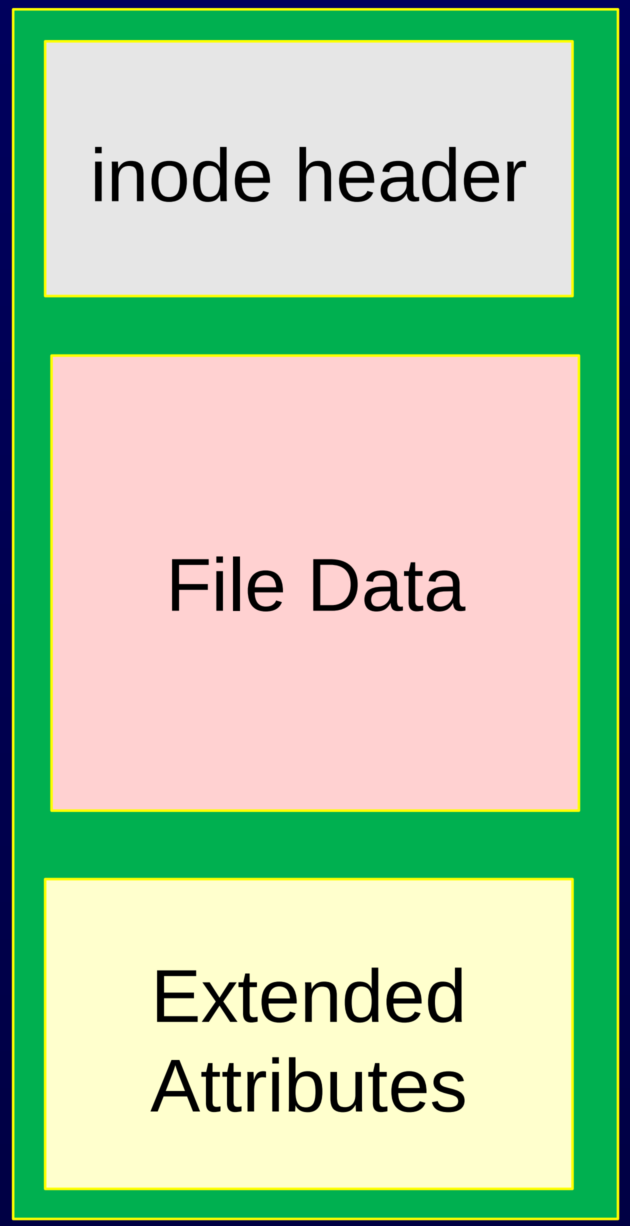


File inode, with Data in inode

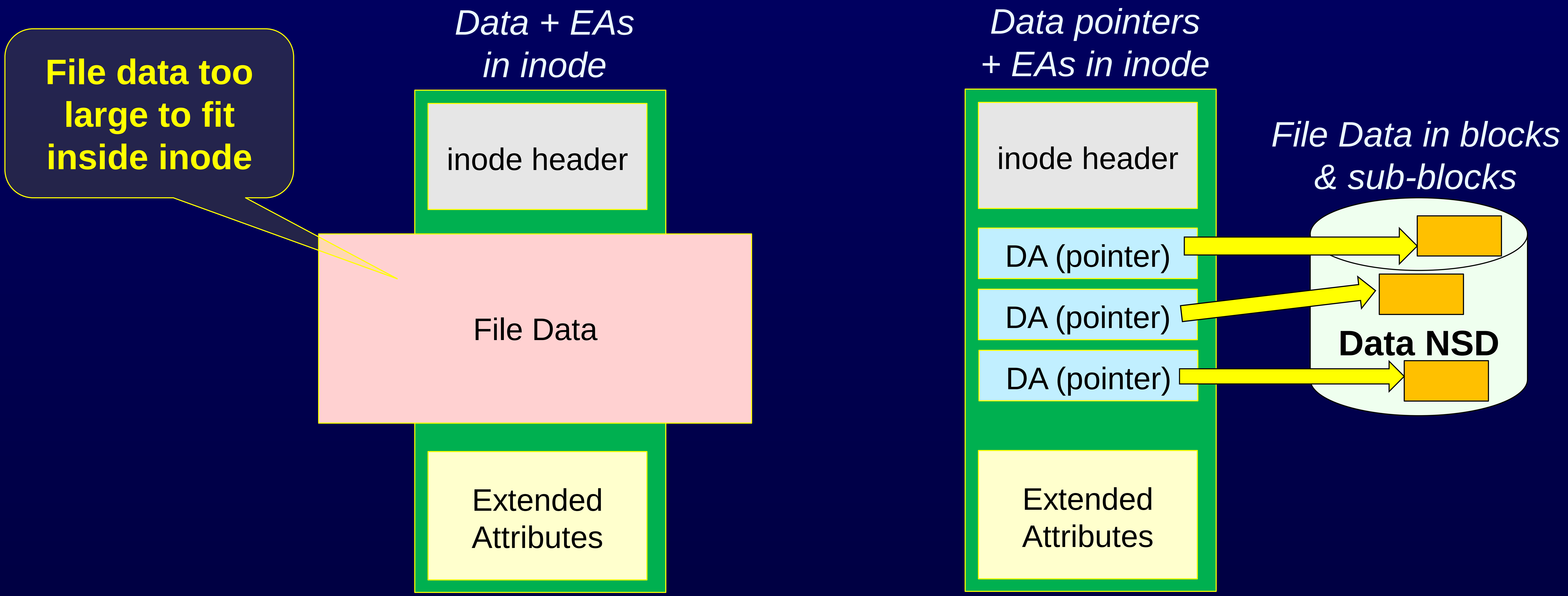
Data in inode



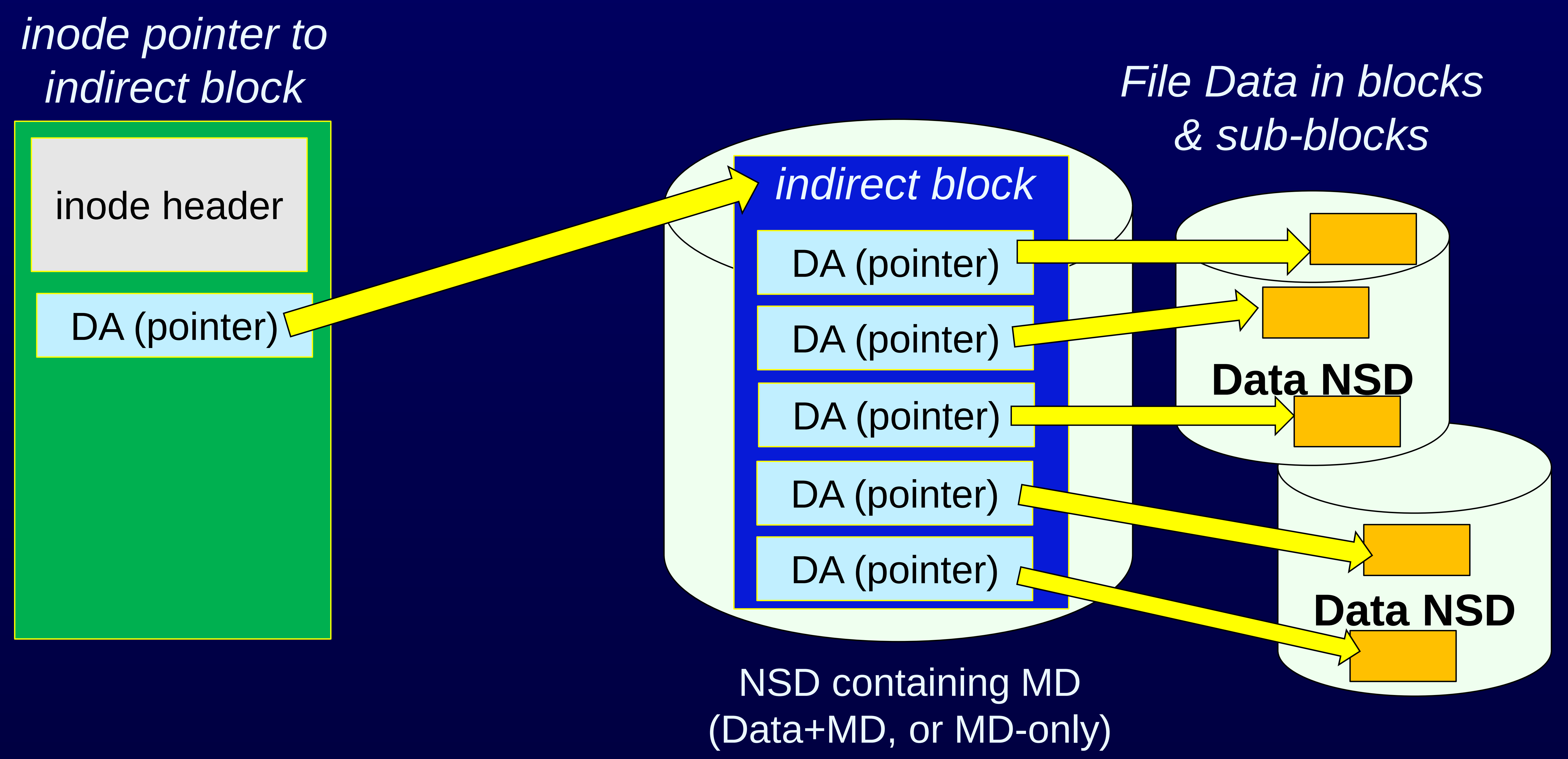
Data + EAs in inode



File inodes containing DAs (pointers to Data blocks)



File inode, with Indirect Block containing DAs



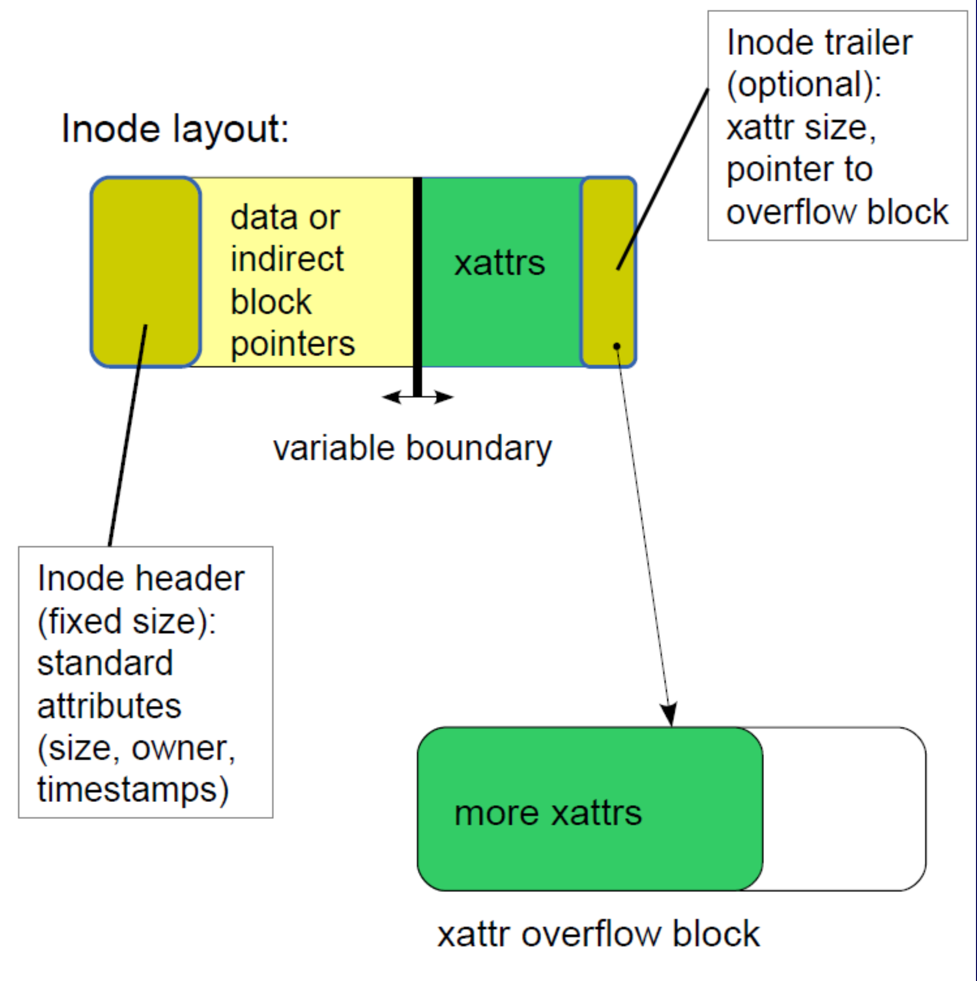
Indirect Blocks- what are they?

- Used when inode cannot hold enough pointers to disk blocks
  - inode runs out of room for block pointers (DAs) + extended attributes (EAs)
    - Block pointers are known as Disk Addresses = DAs
    - Each DA points to a block on disk that contains Data
    - 12 bytes each
  - … and/or too many Extended Attributes (EAs)
    - User defined information, or ILM/Tiering to "offline" tier (tape, object)
      - EAs are not used for "online" SSD⇔Flash⇔Disk tiering

- Can have multiple levels of indirect blocks to support very large files

DAs, inodes, and Indirect blocks

- File’s block pointers (DAs) in an inode
  - More blocks get allocated ⇨ More DAs in the inode
- inode fills up, then add another block ⇨ another DA
  - Indirect Block is allocated
  - Existing DAs copied from inode to the Indirect Block
  - DAs in inode are replaced with one DA pointing to the Indirect Block
    - When the 1st Indirect Block fills up…
      - A new Indirect Block is allocated
      - A new DA in the inode points to the 2nd Indirect Block
      - Can have multiple levels of indirect blocks to support very large files

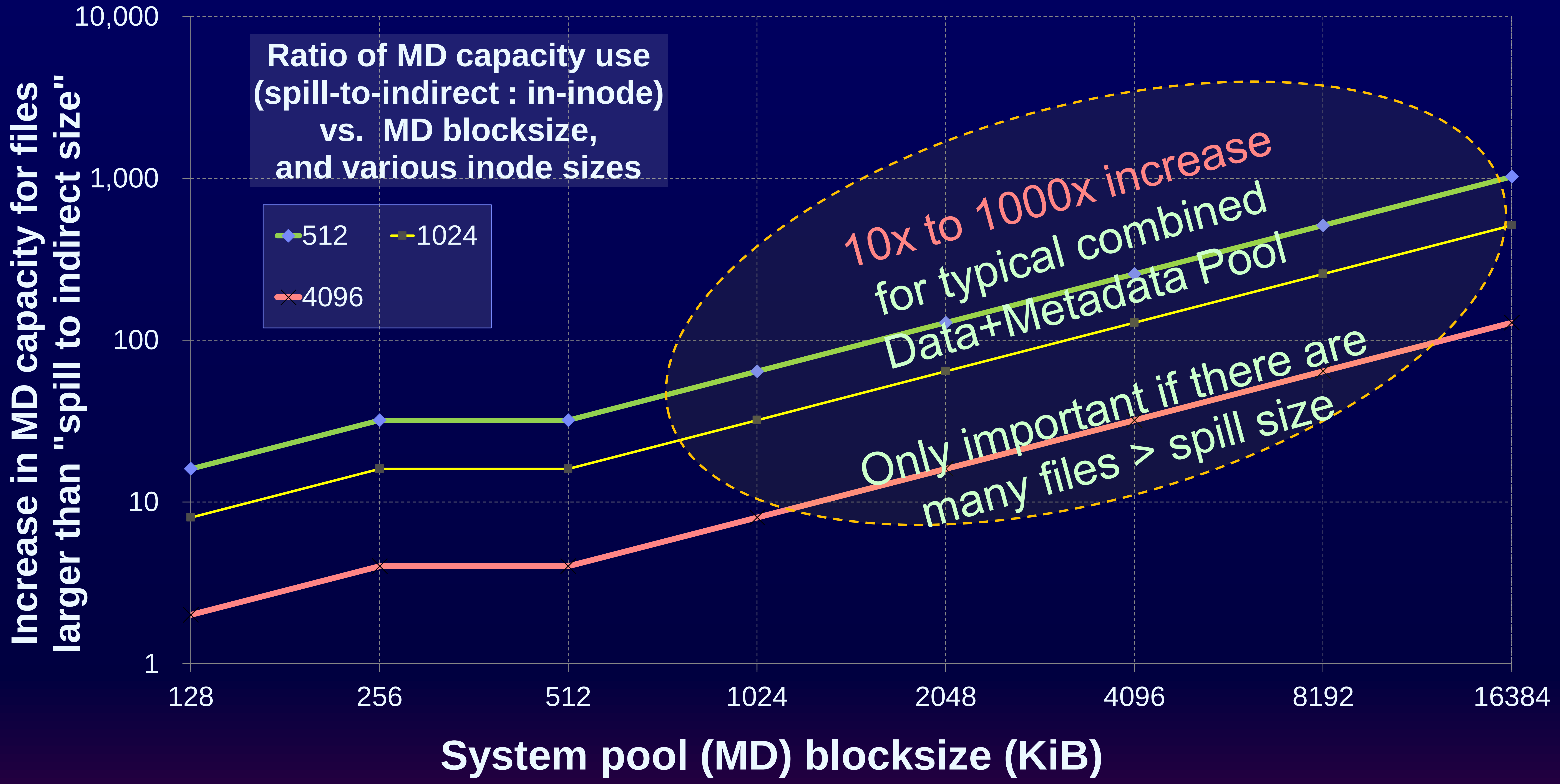


Indirect Blocks and MD capacity

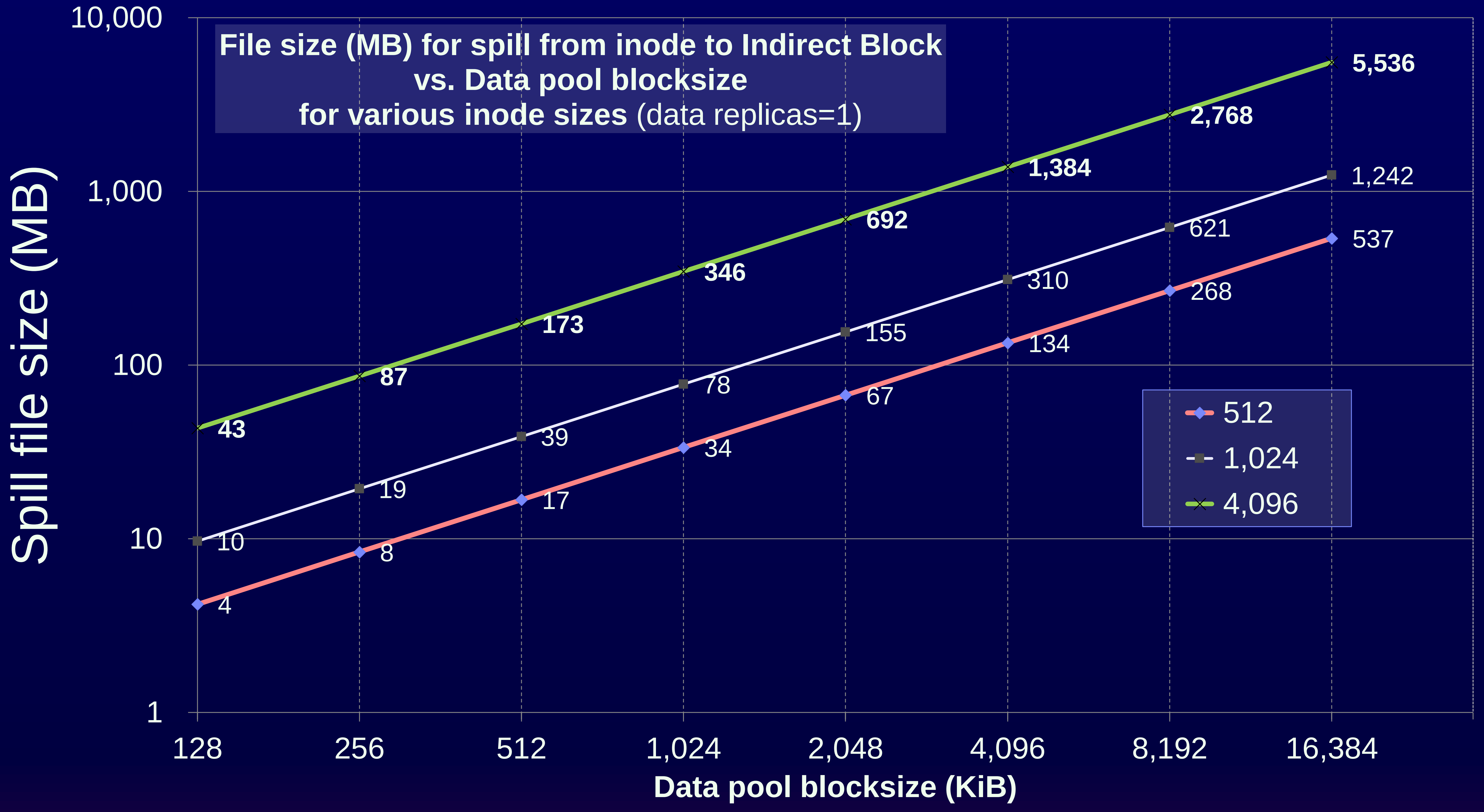
- Too many file blocks = disk block pointers ⇨ DA pointer spill from inode to Indirect Block
- Indirect Blocks can be expen$ive, uses **2x ⇨ 1024x** more MD capacity
  - e.g. for 4KiB inode can store files up to 330 x blocks in size
    - Filesystem defined with max of 1 data replica, no use of EAs
    - (4096 - header size)/(DA size * MaxDataReplicas)
      = (4096 - 128)/(12 * 1) = 330 DAs
      - 110 if maxDAtaReplicas = 3

- Indirect block size varies, depending on filesystem's MD blocksize in System pool
  - **Always** uses at least 1 sub-block in MD capacity
  - 8 KiB for small blocksizes, 16 KiB for medium, 32 KiB for large blocksizes
  - Can only use 32 KiB, even if sub-block is larger e.g. 512 KiB on a 16 MiB blocksize

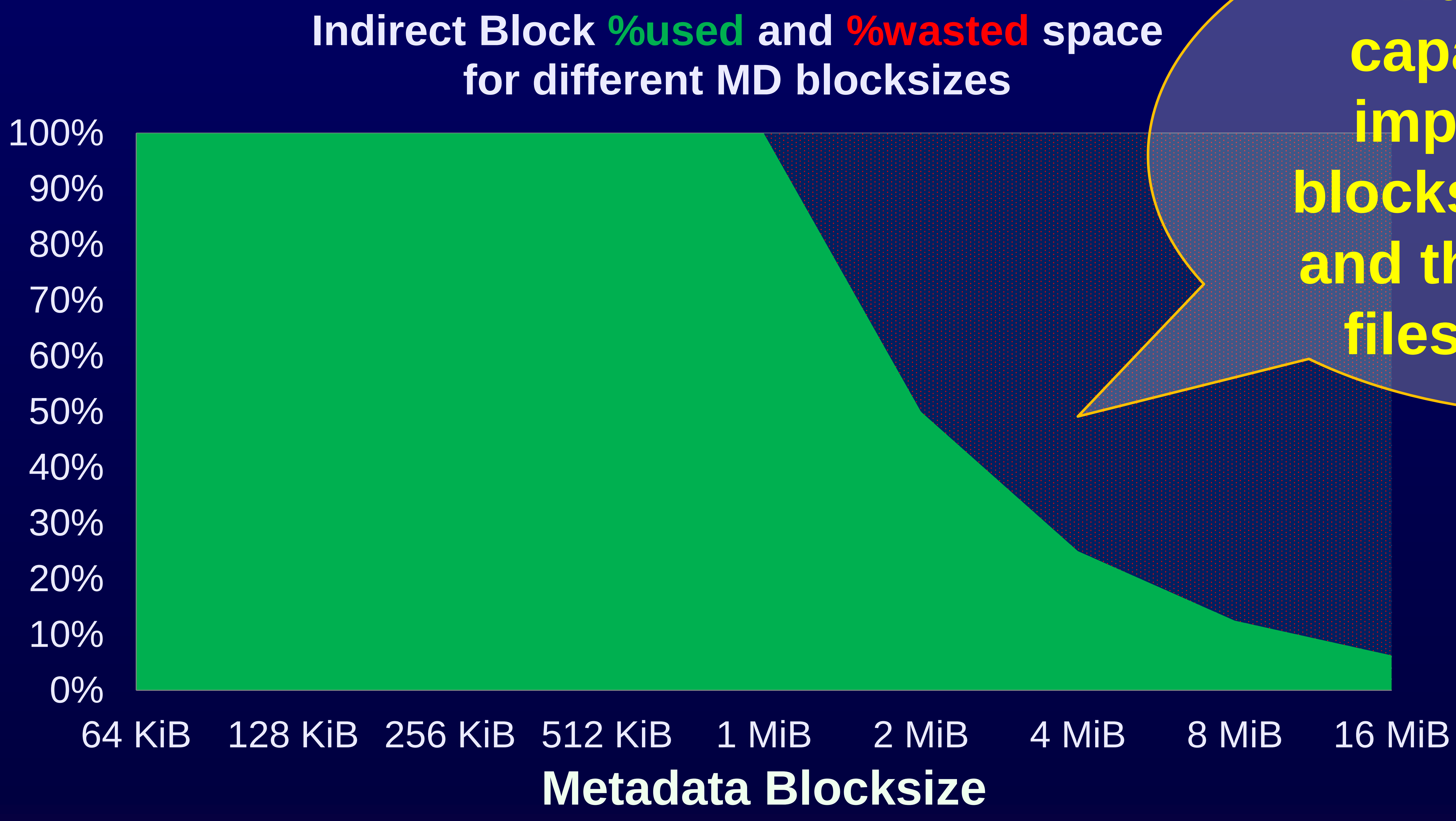
Spilling to Indirect blocks is expensive



Spill-to-indirect file sizes



Indirect Block efficiency



Indrect Block capacity is only important if MD blocksize is v large, and there are many files > spill size!

Indirect blocks and MD blocksize

Filesystem Combined MD & Data blocksize	Sub-block size	Indirect block capacity used by pointers (DAs)	Actual MD sub-blocks used per Indirect Block	Actual MD capacity allocated per Indirect Block	% of allocated MD capacity used	**File size supported by 1 x Indirect Block** (1 KiB inode, combined Data+MD pool)
64 KiB	2 KiB	8 KiB	4	8 KiB	100%	4.8 - 49 MB
128 KiB	4 KiB	8 KiB	2	8 KiB	100%	9.7 - 99 MB
256 KiB	8 KiB	16 KiB	2	16 KiB	100%	19 - 377 MB
512 KiB	16 KiB	16 KiB	1	16 KiB	100%	39 - 755 MB
1 MiB	32 KiB	32 KiB	1	32 KiB	100%	77 MB – 2.9 GB
2 MiB	64 KiB	32 KiB	1	64 KiB	**50%**	155 MB – 5.9 GB
4 MiB	128 KiB	32 KiB	1	128 KiB	**25%**	310 MB – 11.8 GB
8 MiB	256 KiB	32 KiB	1	256 KiB	**13%**	620 MB – 23.5 GB
16 MiB	512 KiB	32 KiB	1	512 KiB	**7%**	1.2 GB – 47 GB

How big are files?

- Mt Sinai School of Medicine, life sciences (genomics) workload

  Big Omics Data Experience, a paper presented at SC15: SC '15, November 15-20, 2015, Austin, TX, USA ACM 978-1-4503-3723-6/15/11. http://dx.doi.org/10.1145/2807591.2807595

  - Numbers of files:
    - 50% of files <= 29 bytes
    - 80% of files < 10 KB
    - 60% < 2 KB
    - 70% < 3.7 KB
    - >50% of files fit within 4 KiB inode

How big are files?

- Science, Engineering, Finance sector customer data, Panasas, 2013
  http://www.panasas.com/sites/default/files/uploads/docs/Whitepapers/SOLID%20STATE%20DRIVES%20AND%20PARALLEL%20STORAGE.pdf
  - Numbers of files:
    - Files up to 64 KiB = 43% to 90%
      - Most sites in survey between 60% to 80%
  - Capacity used, files up to 64 KiB = 0.1% to 2%, most sites<0.5%

Example: 1 PB, files < 1 sub-block

- **4** KiB inode size
  - MD capacity required = 34 TB = 31 TB inodes + 3 TB directories = 3.4% of 1 PB
    - 4 MiB Data blocksize, 128 KiB sub-block size, **4**KiB inode size, 10 files per directory, 1 replica of MD

- **1** KiB inode size
  - MD capacity required = 9 TB = 8 TB inodes + 1 TB directories = 1% of 1 PB
    - 4 MiB Data blocksize, 128 KiB sub-block size, **1** KiB inode size, 10 files per directory, 1 replica of MD

  - If Max Data replicas set to 2, max number of files is ½ x 7.6 B = 3.8 B files
    - Max Data replicas set to 3, max number of files is $^{1}/_{3}$ x 7.6 B

  - If Default MD replicas set to 2, max size of MD is approx 2 x 31 TB

Collecting Metadata info from Spectrum Scale

Existing filesystems can help calculate projected metadata

On a running Scale system, use the *filehist* script to collect some statisics

- filehist does a fast inode scan of all file systems
  - Collects and prints file size and directory occupancy statistics
- It is found in the samples directory:
  /usr/lpp/mmfs/debugtools/samples/filehist
- Running filehist requires the tsinode utility:
  /usr/lpp/mmfs/samples/util; make tsinode

Data and MD replication

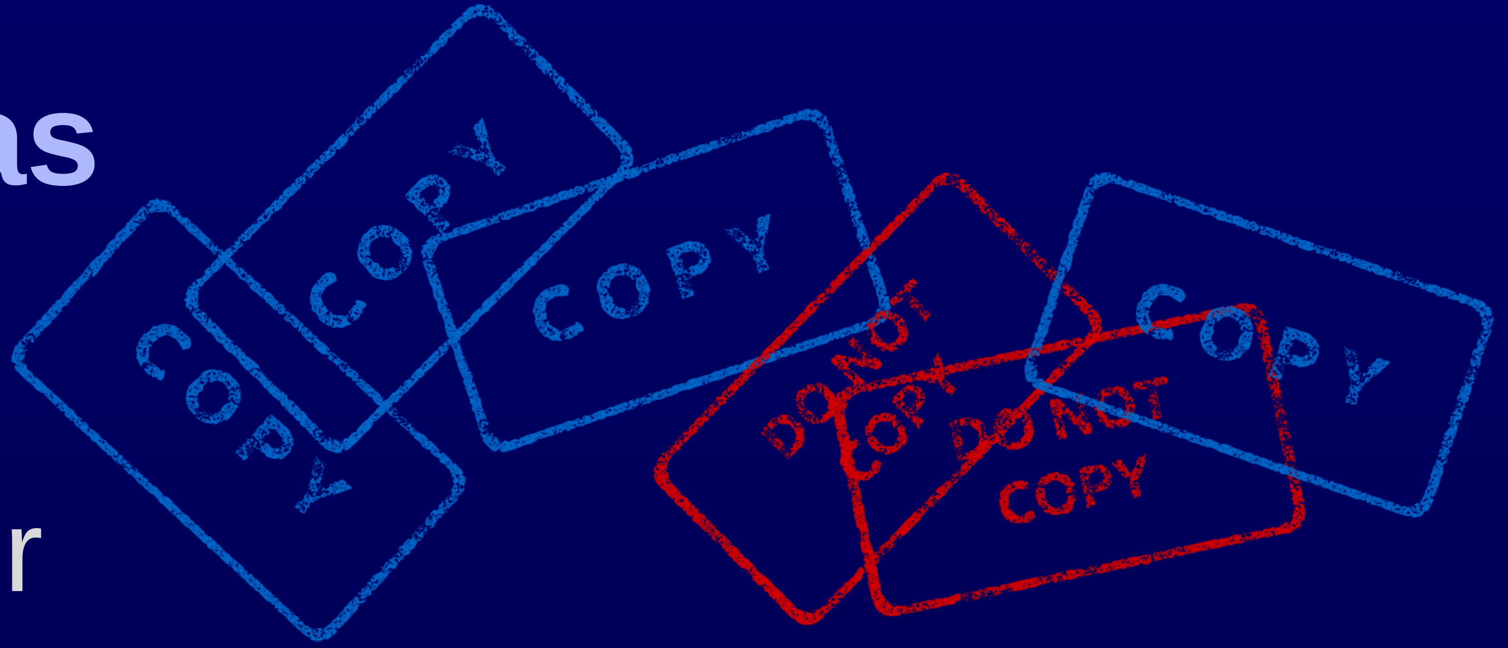
- How many replicas = copies of information
  - Maximum = 3 = 1 + (2 copies) 🡐 Note: there is no “master” copy!
  - Metadata replicas = inodes, directory blocks, indirect blocks etc.
  - Data

Data and MD replication: maximum replicas

- **Max** replication settings for Data and Metadata
  - Max can **only** be set at filesystem creation *(mmcrfs)*
  - Max replicas for MD has little effect on MD capacity used, until the default is changed to >1 (TBC!)
    - And *mmrestripefs -R* is run
  - **Max replicas for Data, multiplies the MD capacity used**
    - **Reserves space in MD for the replicas even if no files replicated!**

- **Default** replicas of MD and Data
  - Set at filesystem creation, can change later
    - *mmchfs … -m DefaultMetadataReplicas … -r DefaultDataReplicas*
  - Number of data replicas does not have to be the same for all files in a filesystem
    - You can set data replication on a file by file basis using policies or *mmchattr*

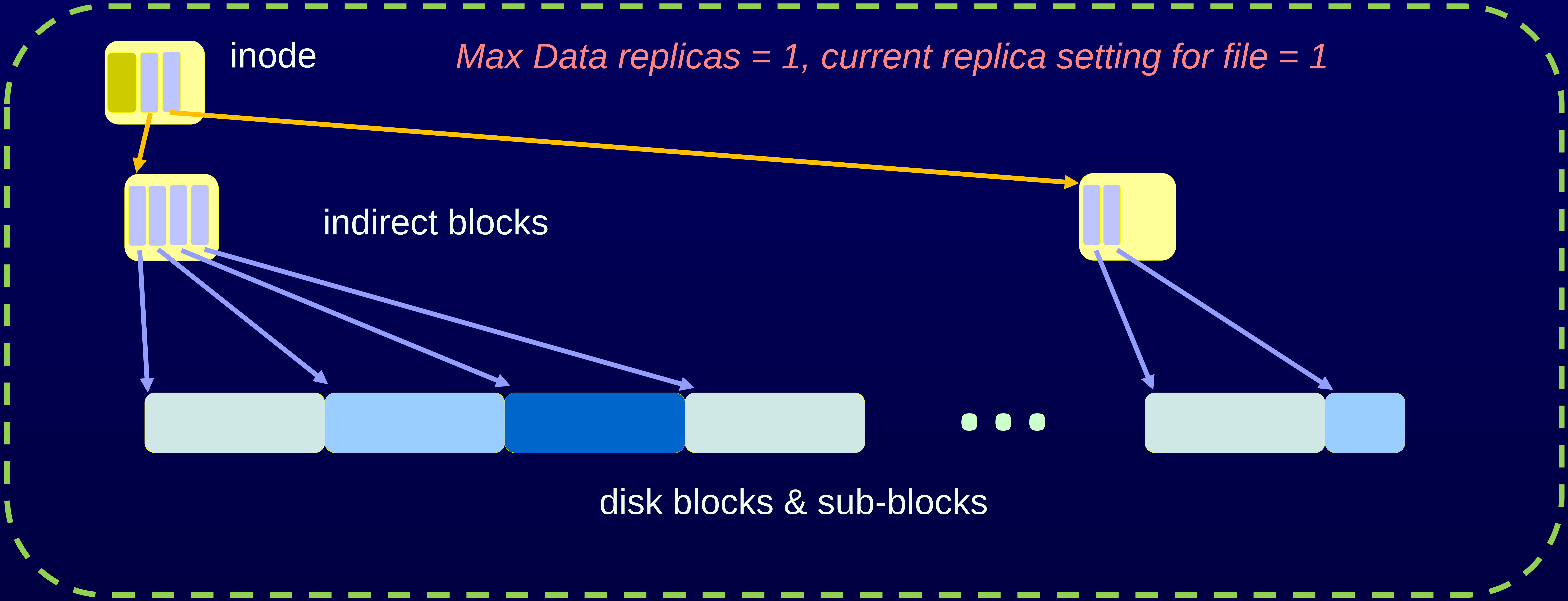
Data and MD replication: default replicas



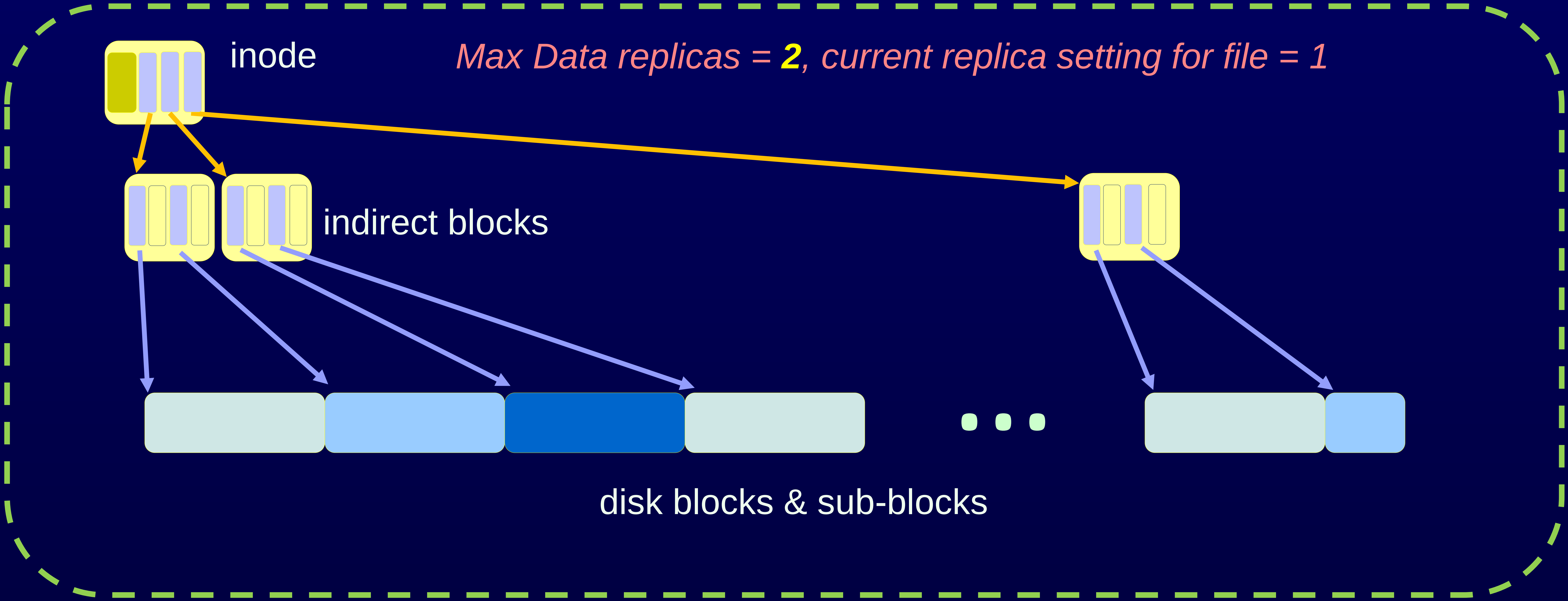
- **Default** replicas of MD and Data
  - Set at filesystem creation, can change later
    - *mmchfs …   -m DefaultMetadataReplicas … -r DefaultDataReplicas*

- Number of data replicas does not have to be the same for all files in a filesystem
  - Can override default replication on a file by file basis
    - Use policy engine or *mmchattr*

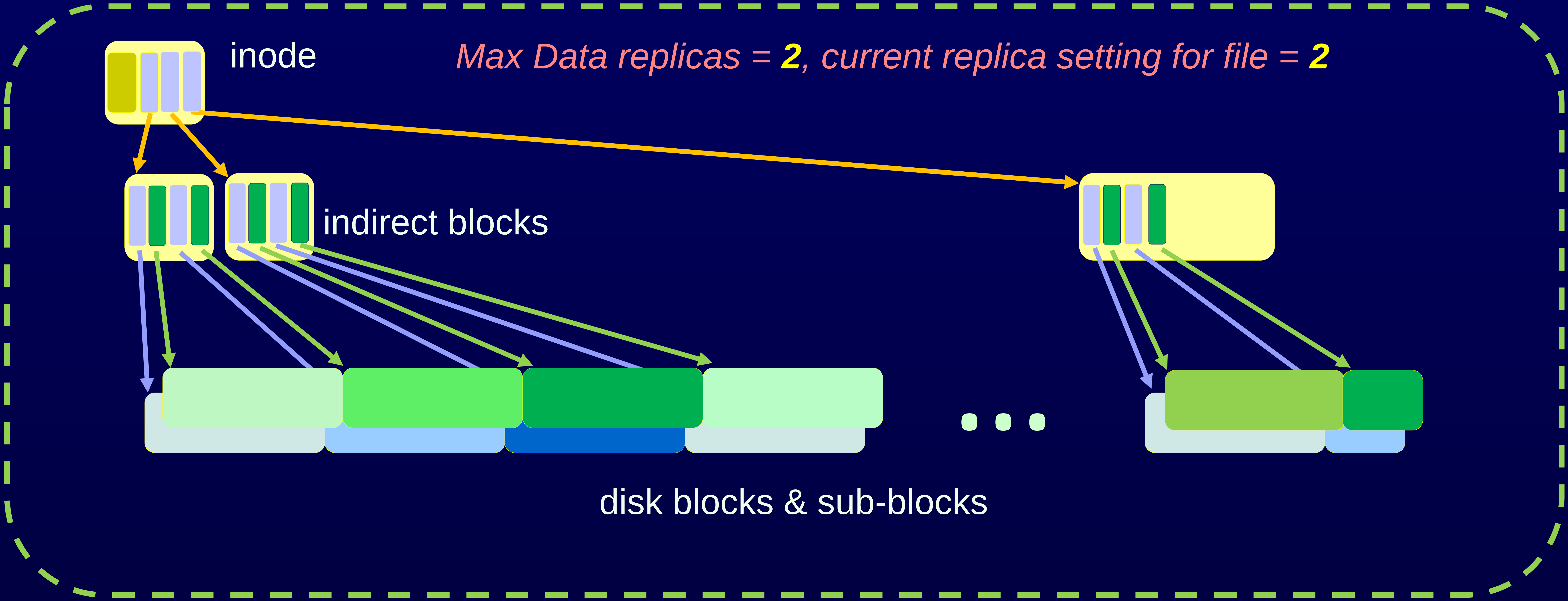
Replication (animation)



Replication (animation)

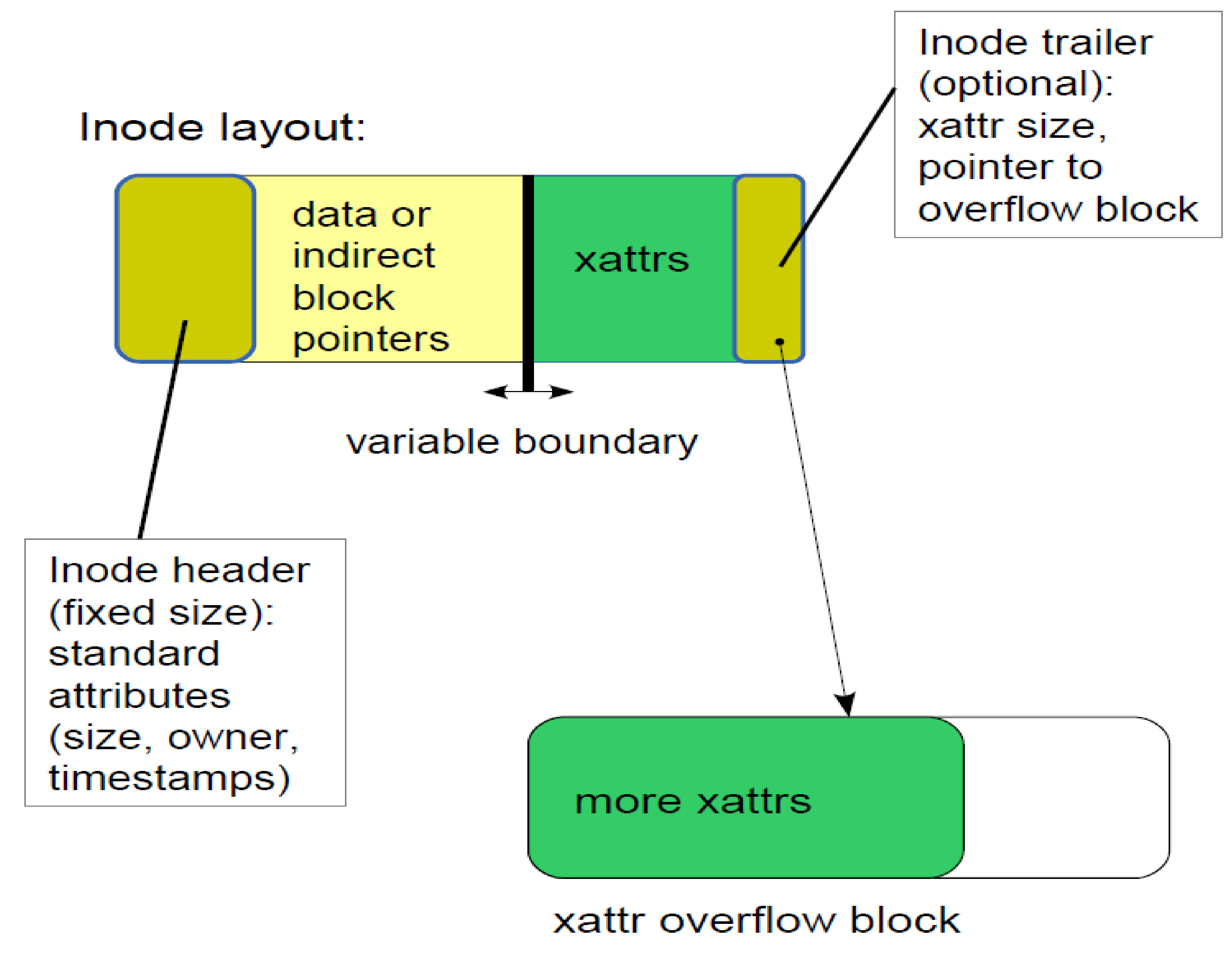


Replication (animation)



EA blocks

- If EAs don’t fit into inode
- inode has a single pointer to EA block
  - Max 64KiB



Directories

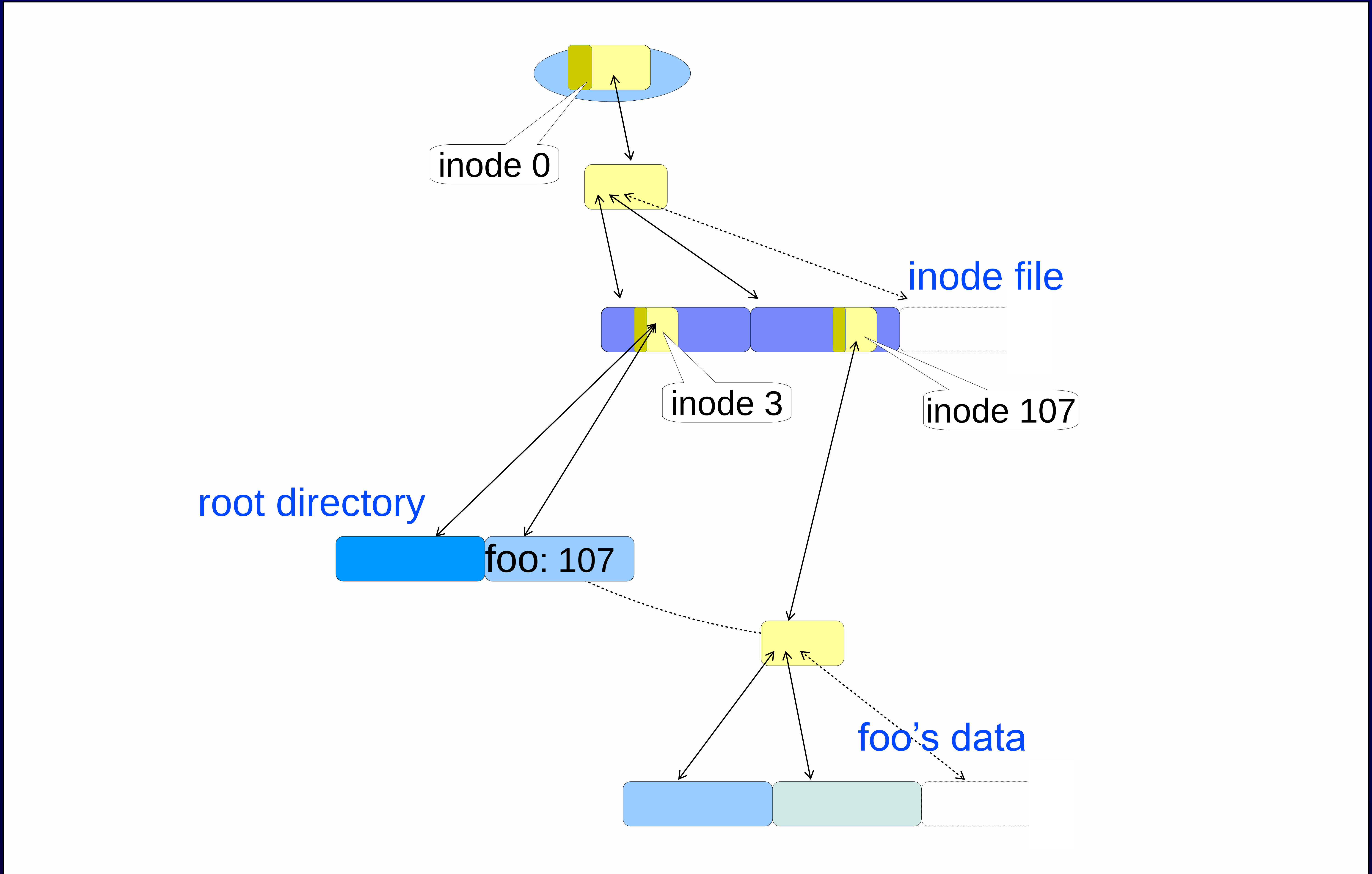
- Each directory is a sparse file
- File starts as 1 sub-block (MD space), grows as required
  - Hashed directory structure ensures constant lookup time
- inode can point to a directory “file”
  - Directory entry for a filename points to inode for that file

Optimizing Space for Directories

Directory inodes can contain file or dir name info to save space

- Embeds file/dir name + pointer to file/dir inode in directory inode
- 512 byte directory inode can contain 12 x 32 byte file entries
    - (512 -128)/32 = 12
    - For 1 KiB inode: (1024 – 128)/32 = 24 file entries
- The average directory size ranges from 10-16 entries, so this is a useful optimization
- 1st 32 byte entry allows names up to 20 bytes (32 – 12 byte header)
    - Longer names take up additional 16 byte blocks

Metadata



inode 0
inode file
inode 3
inode 107
root directory
foo: 107
foo's data

Snapshots and MD

There is no way to accurately predict the effect of Snapshots on MD capacity, except by observation!

- NOT Just # of snapshots x % of data changed!

- Data change % is small, but might touch many files & blocks
  - Many data blocks changed ⇨ Many inodes & indirect blocks changed
    - All changed inodes and indirect blocks have to be copied!
      - Single inode changed ⇨ whole “block” of inode file is copied
      - 16 MiB blocksize filesystem = 0.5 MiB indirect block

- Filename changes ⇨ Directory inodes & directory blocks changed
  - Whole directory copied, even for a single change

When do I need Flash/SSD for MD?

- Bottom Line: When pagepool regularly does not contain the needed inode entries, retrieving metadata from Flash storage makes sense.

- If you use a single storage pool for Data and MD, h/w does not provide enough I/O for both simultaneously
  - Or MD and Data NSDs are on separate pools and LUNs, but share physical disk drives!

Workloads which might need Flash/SSD for MD

- Intensive use of the Spectrum Scale policy engine
  - Spectrum Protect Incremental backups (mmbackup),
  - ILM/tiering: disk⇔Flash/SSD, disk⇔tape, etc.
- Snapshots- deletes of a “middle” snapshot in a series
- Lots of “find”, or “create file”, or “delete file” tasks, esp from OS
- Work on small files with data in inodes

Flash/SSD: "I am tired of your constant writing!"

- Run out of pre-cleared Flash areas = write performance drops
  - "Pre-conditioning curve", present in SSDs and some Flash
    - >50% drop in IOps after 0.5-1 hour of continuous operation
      - at 70% Read 30% Write (8K)

  - Can delay drop in performance by designing SSDs/Flash with more spare capacity
    - = More $$$, less usable capacity!
    - Note that IBM Flashsystem contains unique features to reduce "write fatigue"

Source: storagereview.com review of Intel 2TB P3700 SSD, Dec 2015
Note: this is a good result!

MD Recommendations

Recommendations

- Understand the workload!
  - e.g. Is it OK to have 1 set of disks shared between Data and MD i.e. share disk IOPS between Data and MD work?
    - 1000 x Data disk drives
    - One large parallel job that does file creation, then starts work
      - = No Data I/O when I want to do MD I/O
      - = No MD I/O when I want to do Data I/O
    - MD work can use all the IOs of the Data drives... maybe we don't need SSD/Flash!

General **recommendations**

- Do not pre-allocate more than 25% of MD-only pool to inodes
  - Need room for indirect blocks + directory blocks + EA blocks
  - For combined Data and Metadata, pre-allocate small %

- Use MetaData-only NSDs with a small blocksize e.g. 128K, 256K
  - Makes indirect blocks small = space and performance efficient

Choosing inode size

Inode size	Max file data in inode (bytes, no EAs)	Max file size, inode only, 1 MiB Data blocksize 1 replica	Max file size, inode only, 16 MiB Data blocksize 1 replica	MD capacity used compared to 512 B inode (approx.)
512 B	<384	33 MB	500 MB	1x
1 KiB	<896	75 MB	1.2 GB	2x
4 KiB	<3896	340 MB	5.5 GB	8x

Choose 512 Byte inodes?

- Choose 512 Byte inodes:
  - Many zero length or v small files
    - Up to ≈ 390 bytes
    - No use of EAs, 1 replica i.e. no add’l copy of data
  - Many large files, that would overflow into Indirect Blocks anyway

1 KiB inode size?

- 1 KiB inodes: “the under-rated middle sibling”, up to 75% less MD space than 4K inodes
  - May be a good compromise between 512 and 4 KiB
    - Data in inode: Files ≈ 700 bytes
    - Files ≈ 100 MB to 1 GB without using Indirect Blocks
      - 34 MB for fileystems with 1 MiB blocksize, 1.24 GB for 16 MiB
        - No use of EAs, 1 replica i.e. no add’l copy of data
  - But if have many large files > 1 GB, 4KB might be better
    - No use of indirect blocks

4 KiB inode size?

- 4 KiB inodes
  - Files of up to ≈ 3.5 KB (approx) in inode
  - Files of ≈ 0.5GB to 5 GB without using Indirect Blocks
    - 346 MB for fileystems with 1 MiB blocksize, 5.5 GB for 16 MiB
      - No use of EAs, 1 replica i.e. no add’l copy of data

Recommendations: analysis of MD

- **For well known file sizes, can “tune” MD and other parameters to match**
- Calculate, play with block sizes and inode sizes
  - *Could we enable max data replicas =2 or 3 without MD cost?*
    - 2x or 3x the number of DA pointers might still fit into inode or Indirect Block
  - *Could inodes be larger or smaller to save on capacity?*
    - Larger inodes might avoid expensive “spill to Indirect Block”
    - Smaller inodes if files are so large they are must use Indirect Blocks
  - *Larger or smaller Data blocksize make the capacity use more efficient?*
    - Performance impact? Space efficiency for data?
  - *What if I get the file sizes 10% wrong, or the future use changes?*

MD Recommendations: Storage

- MD on separate physical storage- a good idea or not?
  - Spectrum Scale is good at caching MD for both read and write
    - But if have a lot of random MD access, Flash/SSD is good
      - Random = normal *find* and other commands, or other work
  - Workload patterns- interference between MD-intensive and Data intensive work?
    - If little interference, maybe share disk storage… 1000 disks is a lot of IOPS!
  - Would SSD/Flash be useful?
    - MD intensive work
      - e.g. directory scans, policy engine, incremental backups, snapshots…
    - Intensive work on “data in inode” files

MD Recommendations: LROC and HAWC

- LROC=read caching in SSD on Client node
  - Can set to cache only Data, only inodes, only Dir blocks, or combination
  - *mmchconfig* options

- HAWC=write caching in SSD on Client nodes (replicated) or central Flash/SSD device
  - Best for lots of small block writes, so probably good for MD
  - New section on HAWC in 4.2 manuals

MD recommendations

- Do not always use 7% or 10% rule-of-thumb for MD allocation
  - Safe in most cases, but… can result in a waste of expensive resources
    - 10% of 4 PB is not required to support 20,000 large files for a TSM storage pool
    - 10% of 4 PB on SSD is 4x as expensive as 2.5%

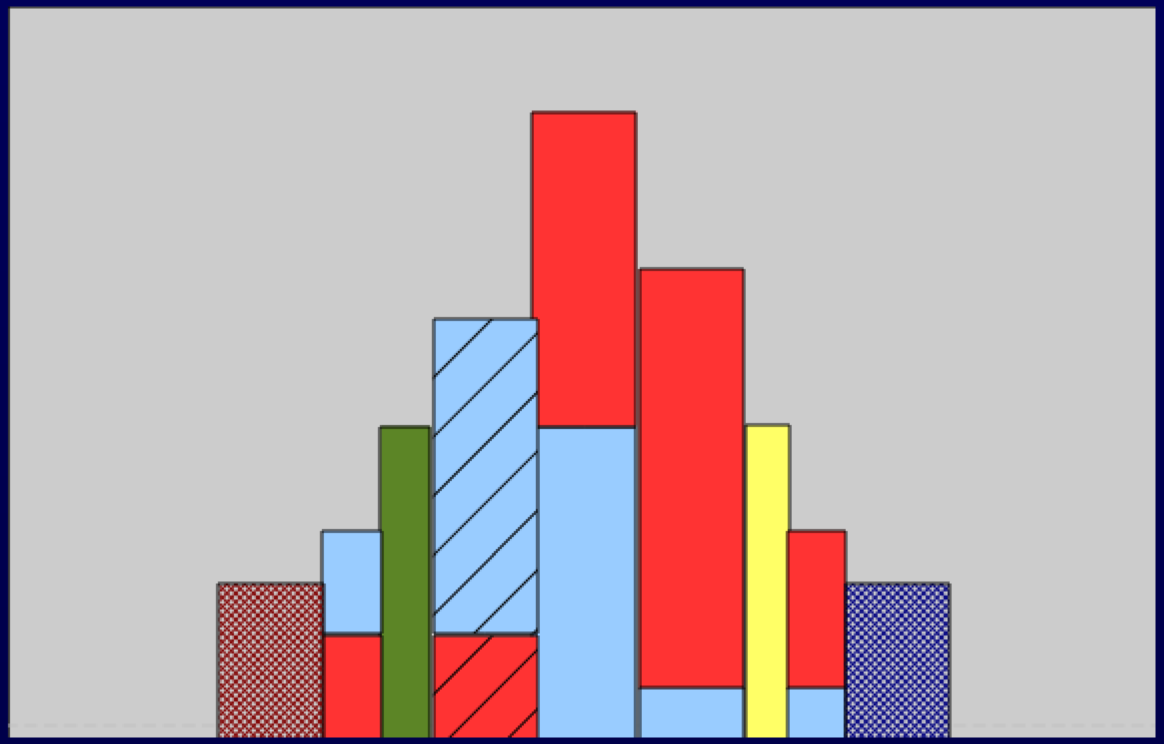
- MD onto a separate set of MD-only NSDs with a small MD blocksize
  - Usually, not always!

- Try to get a histogram of file numbers and capacities
  - I can supply a list of useful numbers for useful "bucket sizes" ≈ 200 buckets!

- Don't panic! About the size of Indirect Blocks… unless you have a lot of files which use them!

mpDIO: multi-process Disk I/O testing using histograms

- **Meta-data testing (and more!)**
- Disk I/O model tool based on histograms
- Not a simulator – actual I/O load
- Portable across operating systems, server hardware
- Parallel
- I/O verification
- Raw device testing
- Randomized read and write with variable block, stride and directional testing
- Directory organization testing
- File lock testing across parallel processes and nodes
- Pre and Post sales technical disk I/O tool



If interested, contact Madhav Ponamgi, *mzp @ us.ibm.com*

Recommendations: RAID

- Storage RAID recommendations:
  - RAID penalty for RAID5/6 and write sizes < RAID stripe size!
  - RAID1 is better for MD, if there will be a lot of writes
    - RAID5 and RAID6 impose more I/O penalty!

Metadata sizing spreadsheet

MD sizing tool (preliminary)

- Very early days!
  - Tool is not finished, not fully accurate, or QAed
  - Has known "simplifications" (and probably some errors)
    - e.g. single level of Indirect Blocks, EAs are not completed etc.
  - **Possibly** useful now! e.g. small number of large files

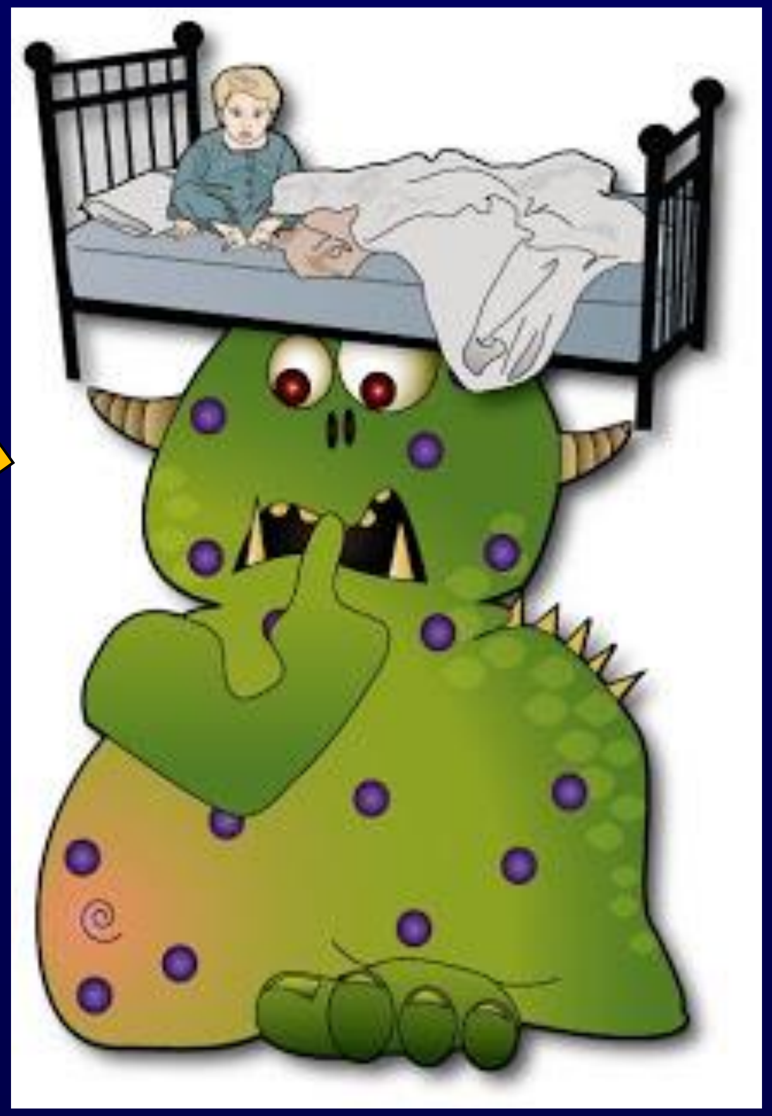
- MD calculations are complex, dependent on many factors…
  - MD and Data blocksizes, inode size, number of directories, use of EAs, replication settings, snapshots, filename length, etc etc etc

MD spreadsheet demonstration

All numbers are in Short Scale (1 B = 10^9) https://en.wikipedia.org/wiki/Long_and_short_scales						
					Units	
Max number of 1xsub-block files possible in filesystem				122.1	Billion	1.2207E+11
Number of files requested				7,629.0	Million	7.6290E+09
One "near-worst case" maximum inode calculation = 1 file per sub-block (inc MD replication setting, includes directory inodes based on 5 files per dir)	27.50%	MD % of data pool		275.0	TB	2.7500E+14
Total MD capacity required from calculations	1.15%			11.5	TB	1.1513E+13
Total inode capacity required from calculations	0.78%			7.8	TB	7.8121E+12
Total directory capacity required from calculations	0.37%			3.7	TB	3.7005E+12
				Only edit **PURPLE** squares	Units	Formulas are here
Total capacity				**1**	**PB**	1.0000E+15
Filesystem blocksize				**256**	**KiB**	2.6214E+05
MD blocksize (if using MD-only System pool, can be different to Data blocksize, often 256 KiB)				**256**	**KiB**	2.6214E+05
MD and data in separate storage pools? (y/n)				Y		
Inode size				**1**	**KiB**	1.0240E+03
Max replicas setting for data (in Spectrum Scale GPFS), set to 2 if using or expecting to use dual site sync clustering				**2**		2
Default replicas setting for metadata (in Spectrum Scale GPFS) 2 for dual site clustering now or future, 1 for single site ESS, 2 for single site non-ESS				**1**		1
Snapshots kept				0		0
% of FS which changes between snapshots				3%		0.03
Extended Attributes used? Includes offline storage pools e.g. TSM, LTFS_EE, etc). Assume that if EAs/ILM is used, can't use base inode for any block pointers				**n**		
ILM/tiering to tape or Object (MCSTORE)?				**n**		
				File class 1		Results
Number of files required				**7.629**	**Billion**	7.6290E+09
File size				**128**	**KiB**	1.3107E+05
Filename size				64	**bytes**	6.4000E+01
Avg files per directory				19	**units**	1.9000E+01
% of total files				100.0%		
Total file capacity requested				909.4	TiB	9.9995E+14
% of total capacity				100.0%		
Total file capacity required	909.4	TiB	9.9995E+14	909.4	TiB	9.9995E+14
Wasted space per file (assuming spread of actual sizes)				4	KiB	4.0960E+03

MD sizing tool plans (preliminary)

- Note: this tool is a “spare time” project (when I should be sleeping)!



- Plans:
  - Phase 1: Finish spreadsheet tool & check against some real Spectrum Scale filesystems
  - Phase 2: Develop scripts that analyse existing filesystems
    - GPFS and non-GPFS filesystems
    - Scripts feed # of files and capacities into tool
    - Allow “what if” analysis, optimise filesystem MD & Data setup based on real data
    - Provide examples from different use cases, industries

The End

Thanks for riding on the trail with me!

Filesets