Give or take

Block and inode allocation in Spectrum Scale

Tomer Perry
Spectrum Scale Development
<tomp@il.ibm.com>

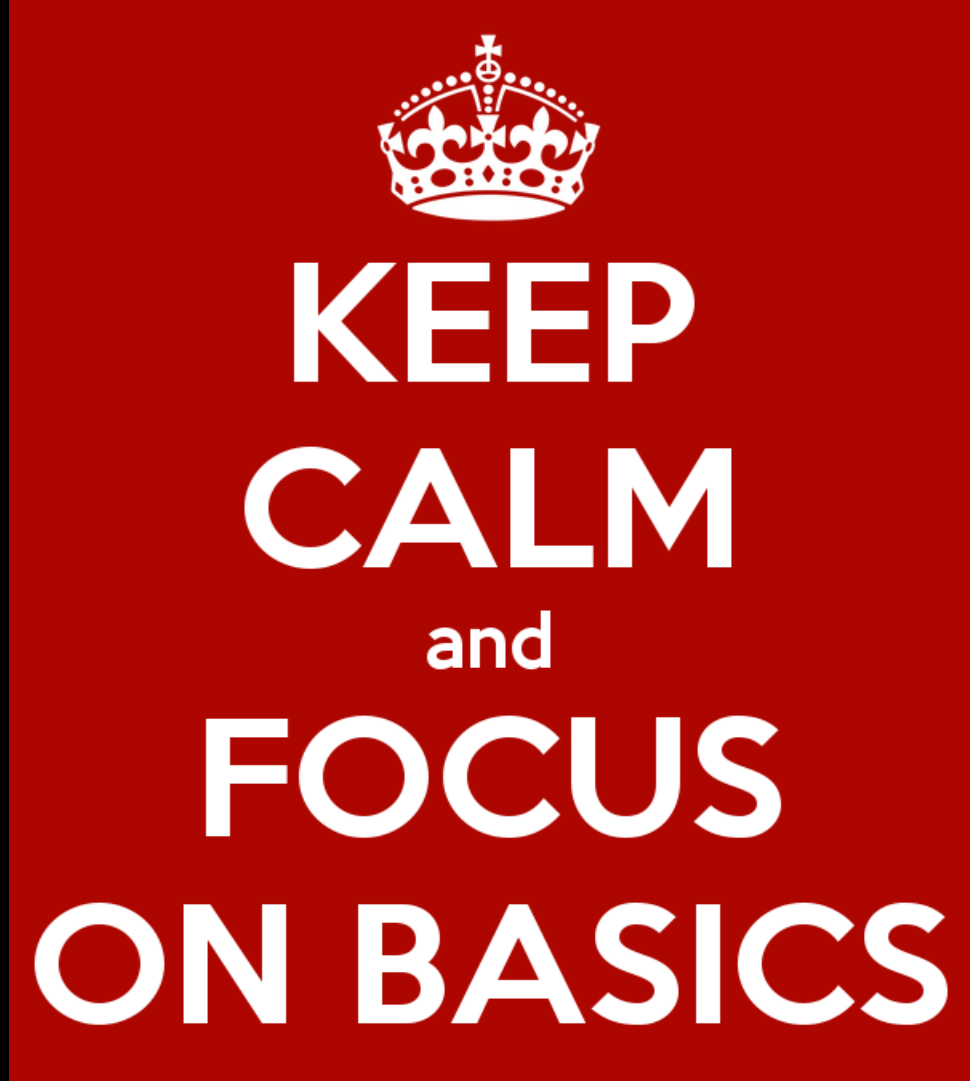
Louise Bourgeois Give or Take 2002

Outline

- Relevant Scale Basics
- Allocation types
- Map based
- Share based

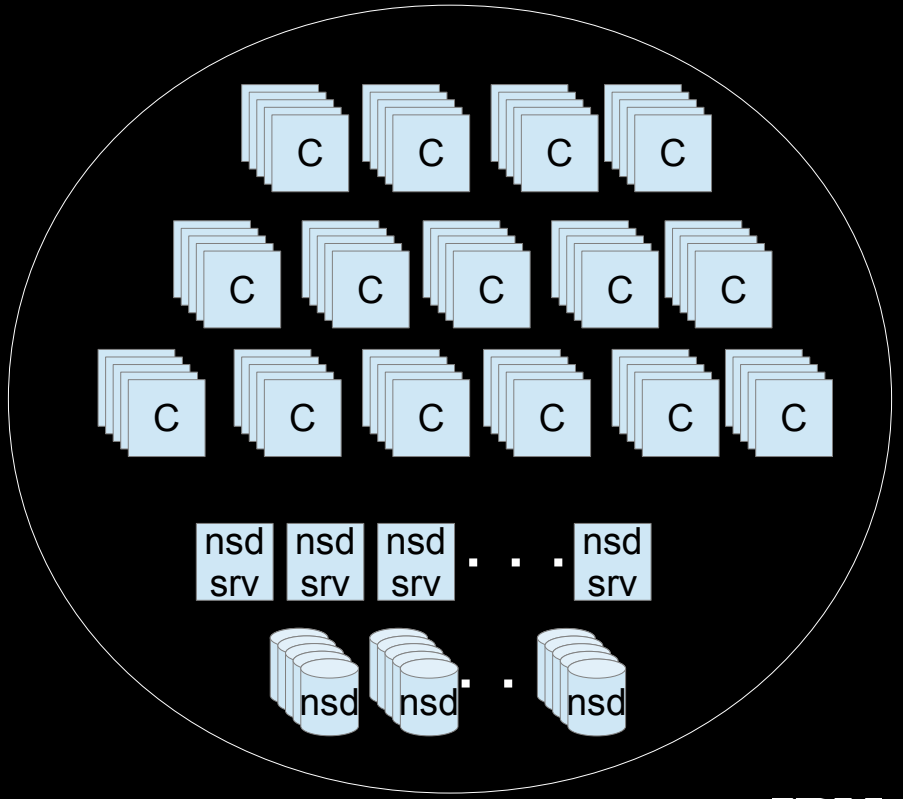
Outline

- Relevant Scale Basics
- Allocation types
- Map based
- Share based



Some relevant Scale Basics
”The Cluster”

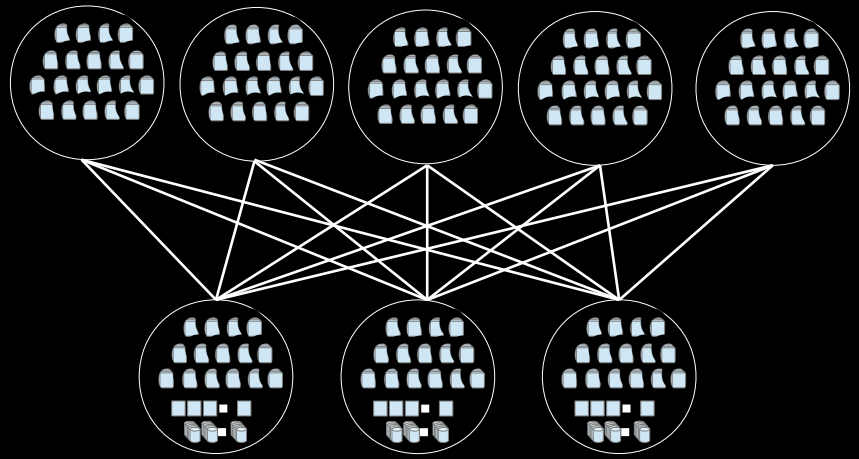
- **Cluster** – A group of operating system instances, or **nodes**, on which an instance of Spectrum Scale is deployed
- **Network Shared Disk** - Any block storage (disk, partition, or LUN) given to Spectrum Scale for its use
- **NSD server** – A node making an NSD available to other nodes over the network
- **GPFS filesystem –** Combination of data and metadata which is deployed over NSDs and managed by the cluster
- **Client –** A node running Scale Software accessing a filesystem



IBM

Some relevant Scale Basics
MultiCluster

- A Cluster can share some or all of its filesystems with other clusters
- Such cluster can be referred to as “Storage Cluster”
- A cluster that don’t have any local filesystems can be referred to as “Client Cluster”
- A Client cluster can connect to 31 clusters (outbound)
- A Storage cluster can be mounted by unlimited number of client clusters (Inbound) – 16383 really.
- Client cluster nodes “joins” the storage cluster upon mount



IBM

Some relevant Scale Basics
Node Roles

- While the general concept in Scale is “all nodes were made equal” some has special roles

Token Manager/s

- Multiple per filesystem
- Each manage portion of tokens for each filesystem based on inode number

Config Servers

- Holds cluster configuration files
- 4.1 onward supports CCR - quorum
- Not in the data path

Cluster Manager

- One of the quorum nodes
- Manage leases, failures and recoveries
- Selects filesystem managers
- mm*mgr -c

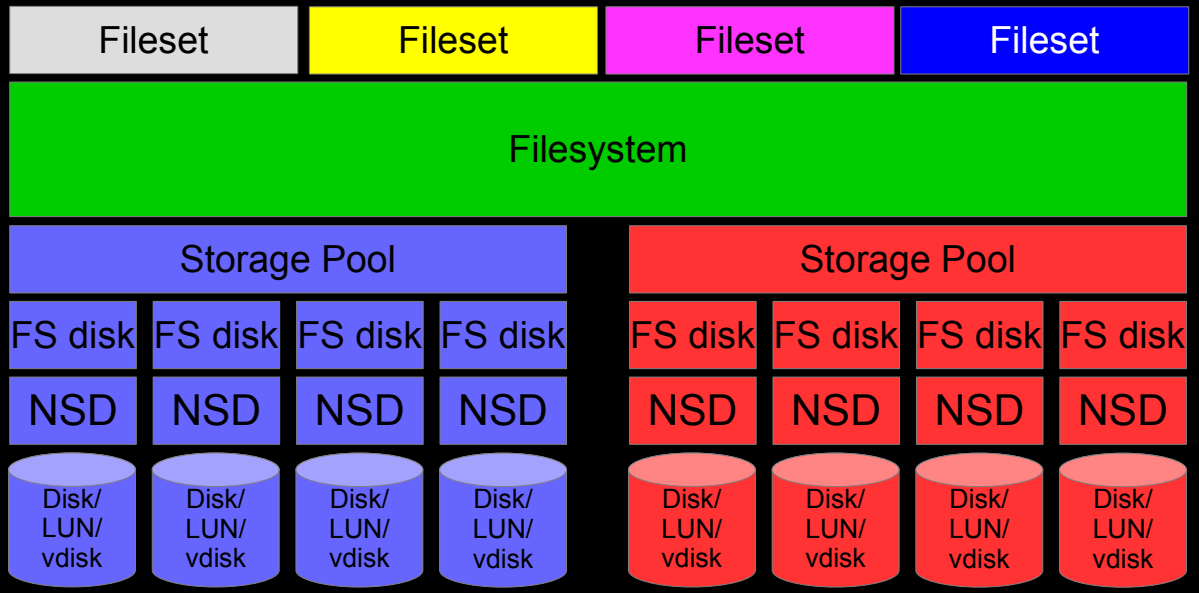
Filesystem Manager

- One of the “manager” nodes
- One per filesystem
- Manage filesystem configurations (disks)
- Space allocation
- Quota management

IBM

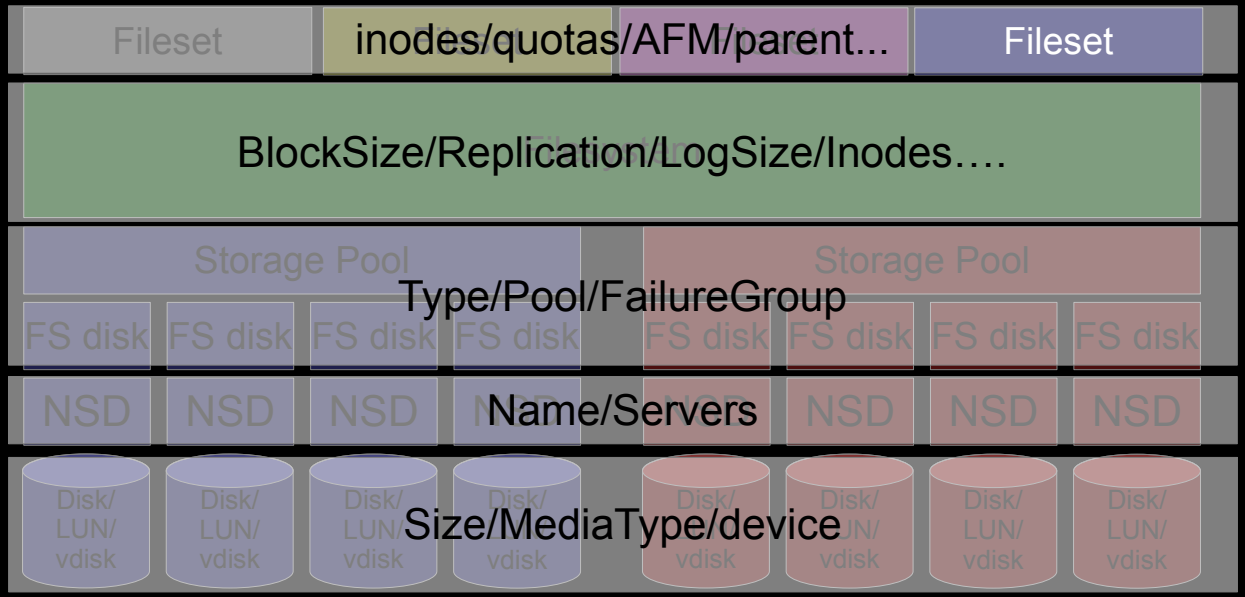
Some relevant Scale Basics
From disk to namespace

- Scale filesystem (GPFS) can be described as loosely coupled layers



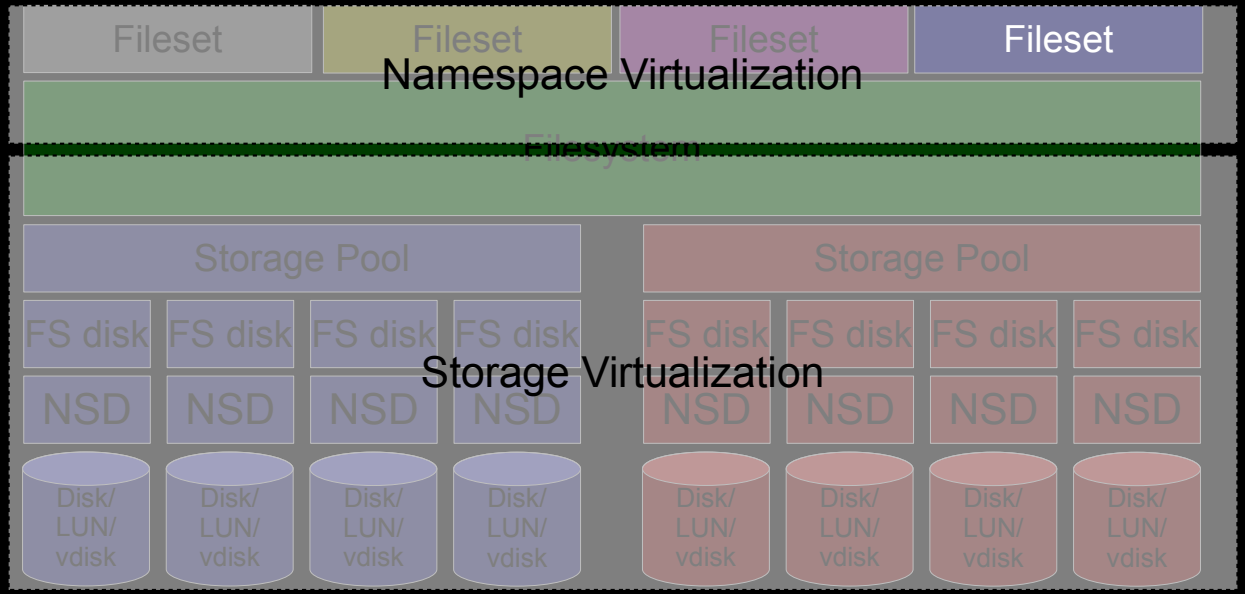
Some relevant Scale Basics
From disk to namespace

- Scale filesystem (GPFS) can be described as loosely coupled layers
- Different layers have different properties



Some relevant Scale Basics
From disk to namespace

- Scale filesystem (GPFS) can be described as loosely coupled layers
- Different layers have different properties
- Can be divided into “name space virtualization” and “Storage virtualization”



Outline

- Relevant Scale Basics
- Allocation types
- Map based
- Share based

Resource allocation
In distributed file systems

- One the main challenges in distributed software in general, and distributed filesystem in particular, is how to efficiently manage resources while maintaining scalability
- Spectrum Scale approach was always “Scalability through autonomy” (tokens, metanode, lease protocol etc.)
- In the context of this session, we will limit the discussion to space allocation and management: subblocks and inodes (essentially, inode allocation is actually “metadata space management”)
- Common approach, taking CAP theorem into account, is to use eventual consistency in order to minimize the performance impact while still providing reasonable functionality.

IBM

Resource allocation
One size doesn’t fit all

- Although there are many similarities between subblock allocation and quotas, there are substantial differences which eventually lead us to use different mechanisms for each

Type	Managed entity	Change rate	Allocation method
Space Allocation	Storage Pools Filesets (inodes)	Adding disks Adding filesets	Map based
Quotas	Users/Groups/Filesets Users/Groups@Filesets	Administrator initiated	Share based

IBM

Resource allocation
One size doesn't fit all

- Although there are many similarities between subblock allocation and quotas, there are substantial differences which eventually lead us to use different mechanisms for each

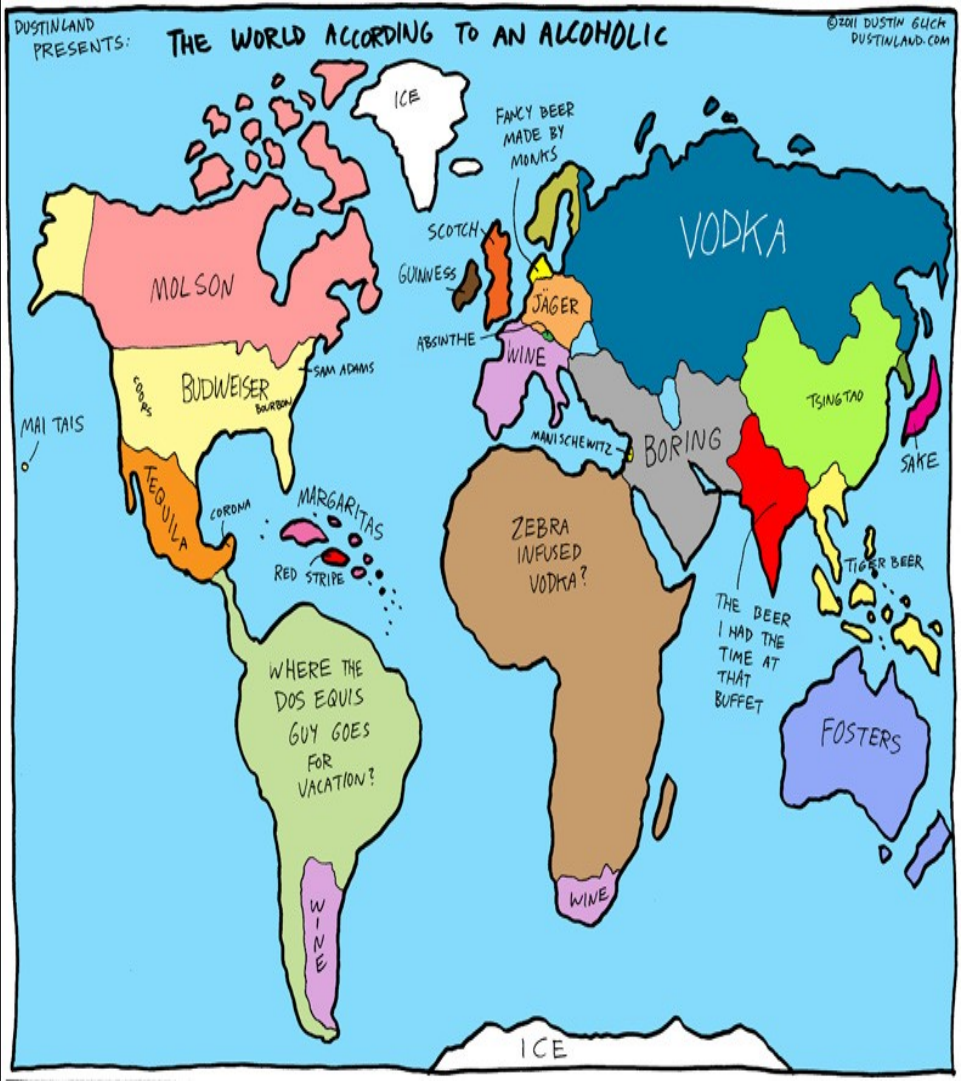
Type	Managed entity	Change rate	Allocation method
Space Allocation	Storage Pools Filesets (inodes)	Adding disks Adding filesets	Map based
Quotas	Users/Groups/Filesets Users/Groups@Filesets	Administrator initiated	Share based

Static

Dynamic

Outline

- Relevant Scale Basics
- Allocation types
- Map based
- Share based

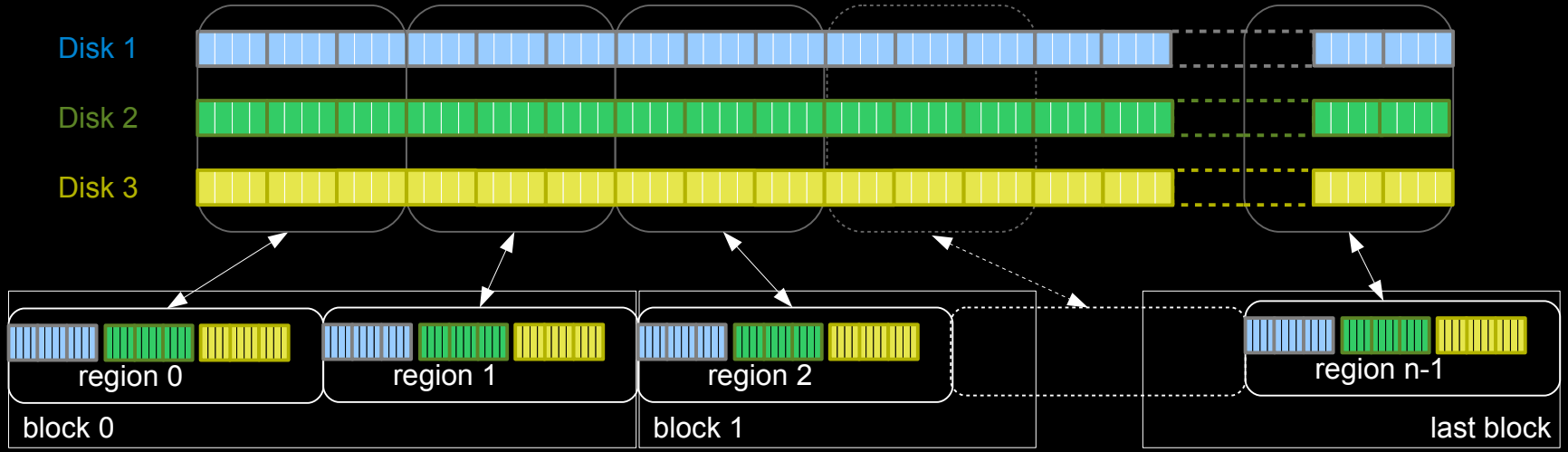


Allocation map based management

- Due to the relatively “static” nature of a filesystem space management it is possible to use the allocation map based approach
- One of the main advantages of this approach is that with proper planning, its relatively easier to allow each node to manage its own allocation independently
- The basic approach is to try and divide the managed resource into “regions”
- When a node needs to resources to be allocated, the filesystem manager assign a region to that node
- From now on, the node can use all the resources in that region
- The number of regions is affected by the “-n” filesystem/pool creation paramter
  (while one can change the -n parameter using mmchfs, it will only apply to pools created after the change)

IBM

Block allocation map



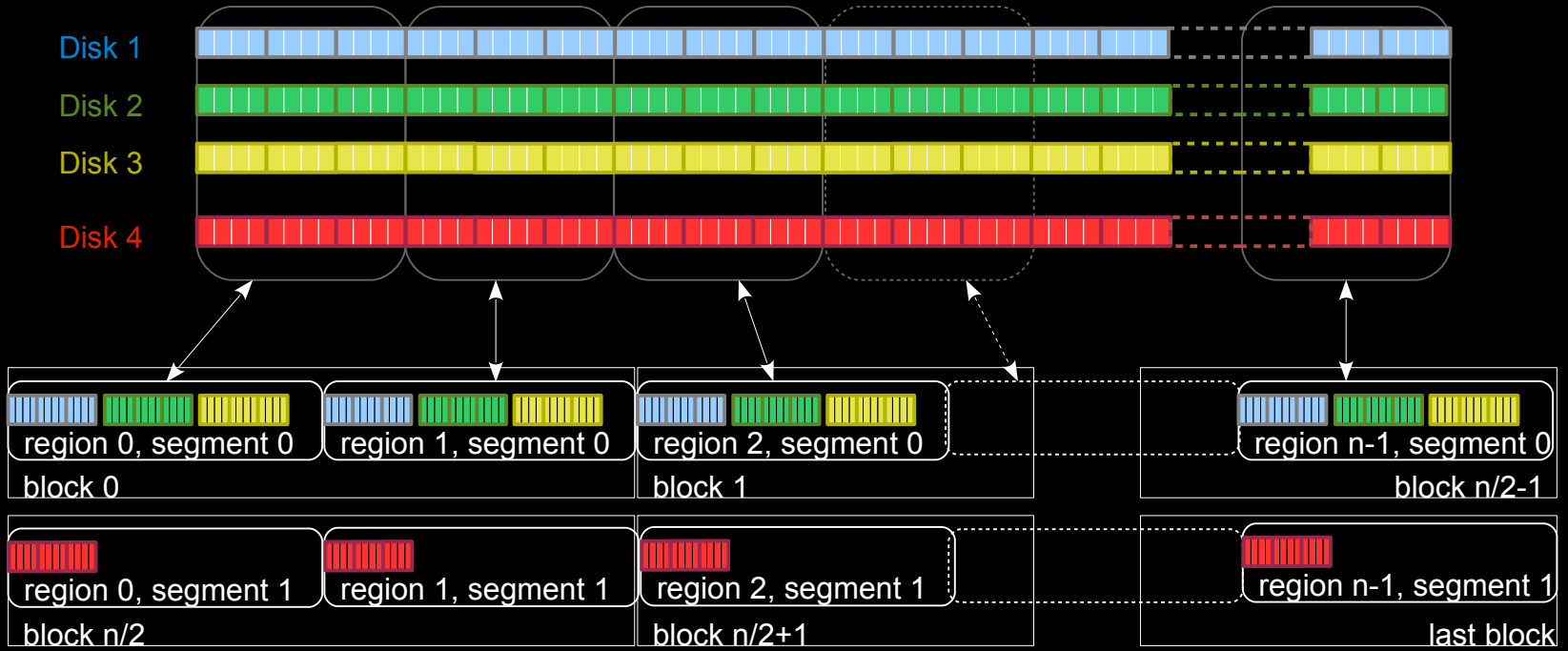
- Divided into a fixed number of n equal size “regions”

  Each region contains bits representing blocks on all disks

  Different nodes can use different regions and still stripe across all disks (many more regions then nodes) – based on the -n parameter on storage pool creation

IBM

Block allocation map



- When adding more disks:

  Alloc map file is extended to provide room for additional bits

  Each allocation region no consists of two separately stored segments

IBM

Inode Allocation map

- The inode allocation map follow the same principal of the block allocation map
- That said, there are differences due to the way inodes are being used:
  - inode have different “allocation states” (free/used/created/deleted)
  - The use of multiple inode spaces (independent filesets) might imply non contiguous allocation, allocation “holes” and other properties that might make management a bit more complex
  - Inode use is usually more “latency sensitive” - thus require using function shipping methods in order to maintain good performance (file creates etc.)
  - Inode expansion might be manual (preallocating) or automatic (upon file creation)
- Inode space expansion logic is outside the scope of this session

IBM

Allocation map

- So what should I use for -n on file system creation?
- Try to estimate the number of nodes that “might” use the filesystem in the future
- Don’t just “use a large number”. The overhead of managing many segments/regions might be constly
- As long as the number is not “way of” - it should be OK.

Outline

- Relevant Scale Basics
- Allocation types
- Map based
- Share based

Quotas 101

- While the main goal of this session is to explain how quotas works in Scale, we should still define some basic quota related terminology
- “A disk quota is a limit set by a system administrator that restricts certain aspects of file system usage on modern operating systems. The function of using disk quotas is to allocate limited disk space in a reasonable way” (source: wikipedia)
- There are two main types of quota: disk quotas and file quotas. The former limits the maximum size a “quota entity” can use while the latter define the number of file system elements that a “quota entity” can use (“quota entity” might be a user, a group or a fileset)
- Other common terms are:
  - Soft quota: a.k.a. warning level. When used with grace period implies real limit
  - Hard quota: the effective limit. A quota entity can’t use more then that limit
  - Grace period: a time frame during which soft quota can be exceeded before it becomes a hard limit

Spectrum Scale quotas

- The dynamic nature of quotas, makes it extremely difficult to balance between flexibility and performance impact
- Thus, an eventual consistency approach was chosen – using “quota shares”
- Basically, the quota manager (part of the filesystem manager role) assign shares to requesting clients. A client can use the share unless being revoked (more details to come)
- Since the quota manager don’t know how much of a given share is actually being used, a new term was introduced: in_doubt.
- The in_doubt value represent the shares that are “out there”. Its a factor of the share size and the number of nodes mounting the filesystem
- In order to get an exact quota usage, the quota manager needs to contact all mounting node in order to revoke the current shares they have.

Spectrum Scale quotas
”The share”

- Since we don’t want a client to contact the quota server for each request, we use the concept of shares (block and file share)
- Every time a node want to allocate a resource, it will contact the server and will receive a share of resources
- In the past, the share size was static (20) but configurable. So one would need to balance between performance impact and functionality (what if in_doubt is bigger then the quota…)
- Nowdays, dynamic shares are being used. It starts with `qMinBlockShare/qMinFileShare` (* blocks/inodes) and might change based on usage patterns

IBM

Spectrum Scale quotas
”The share flow”

- The quota client node will calculate how much blocks/inodes it should request based on historical data and will send the request to the quota server. It might be extremely aggressive with that…
- The quota server will consider the request from a much wider perspective: Number of nodes, remaining quota (-in_doubt), grace period etc.
- The quota server will reply to the client request with the granted resources (blocks or inodes)

IBM

Spectrum Scale quotas
”The share flow” - under pressure

- It is also important to understand what’s going on when we’re getting “close to the limit”. It always takes in_doubt into account as if its used
  - Soft limit exceeded:

    The grace counter kicks in

    SoftQuotaExceeded callback is being executed (if configured)
  - Grace period ends:

    Quota exceeded error
  - Getting close to hard limit:

    Quota server revoke from all clients for each request...performance impact

IBM

Spectrum Scale quotas
tips and tricks

- Sometimes, balancing the impact of quotas might be challenging. For example, imagine and application which all its threads are being “throttled” due to quota revokes slowing down access to other quota objects which are not “close to the edge” (who said Ganesha?).
- In those cases, it might be better to “fail fast”
- There are two major ways to achieve that:
  - Use soft quotas only with short grace period (1 sec for example). In this case, shortly after the soft quotas were exceeded, the app will get quota exceeded error. It will use “a little more then the limit”
  - Use the (undocumented) `qRevokeWatchThreshold` parameter. Every time we’re getting close to `qRevokeWatchThreshold*qMinShare` we need to do a revoke. We will add this revoke to the watch list. We won’t do another revoke to that object for the next 60sec. So, if another share request will come in – we will deny it. This will use “a little less then the limit”

IBM


TWENTY QUESTIONS
DICK
HERB
VAN
FLORENCE
ALDO RAY