IBM Spectrum Scale
User Group
Cologne 2022

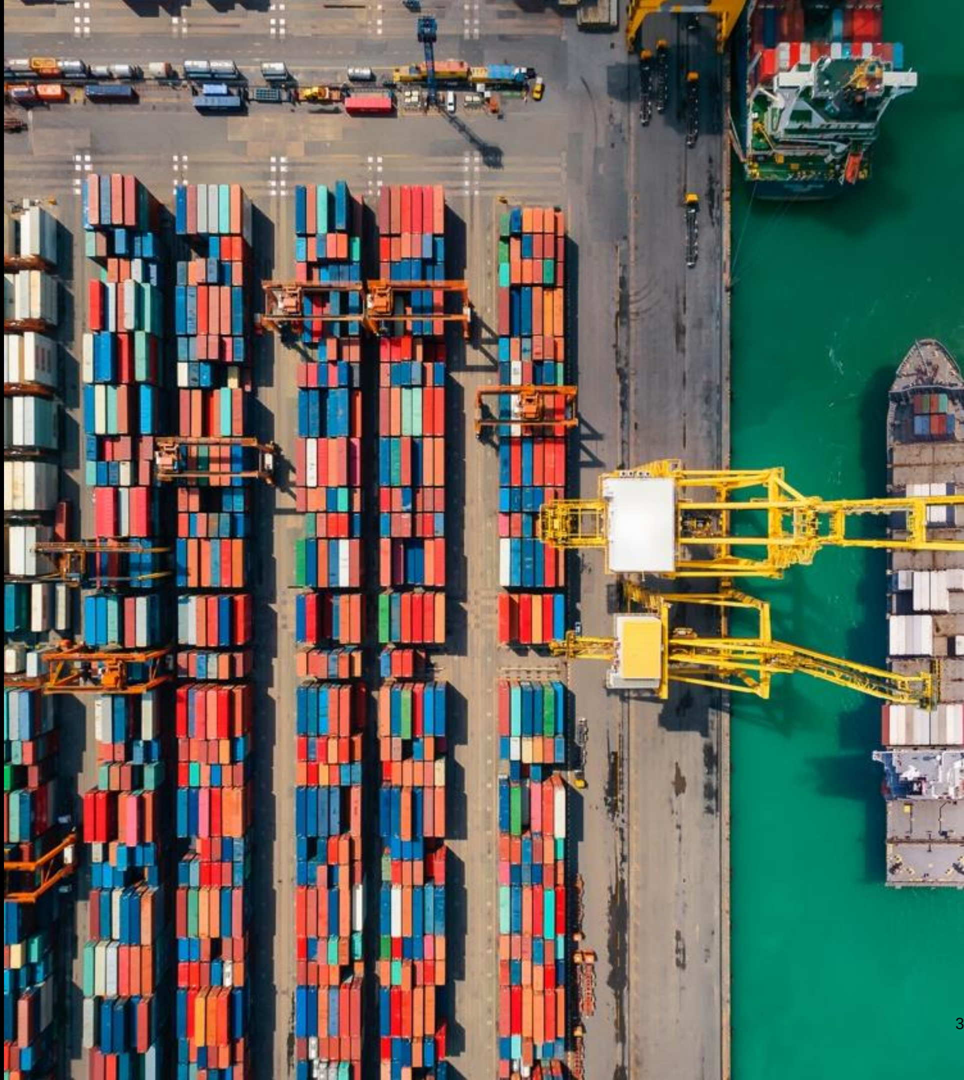
—

Container Native Storage Access Update

Alexander Wolf-Reber

Disclaimer

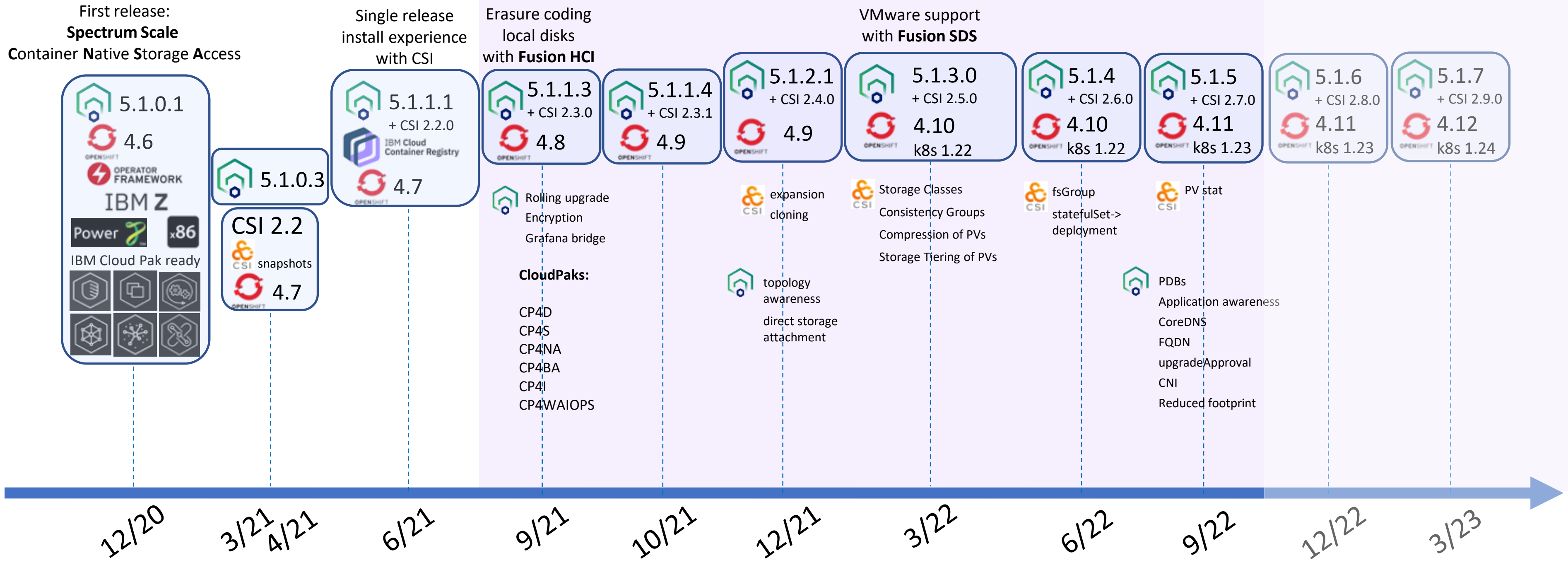
What is new in CNSA?

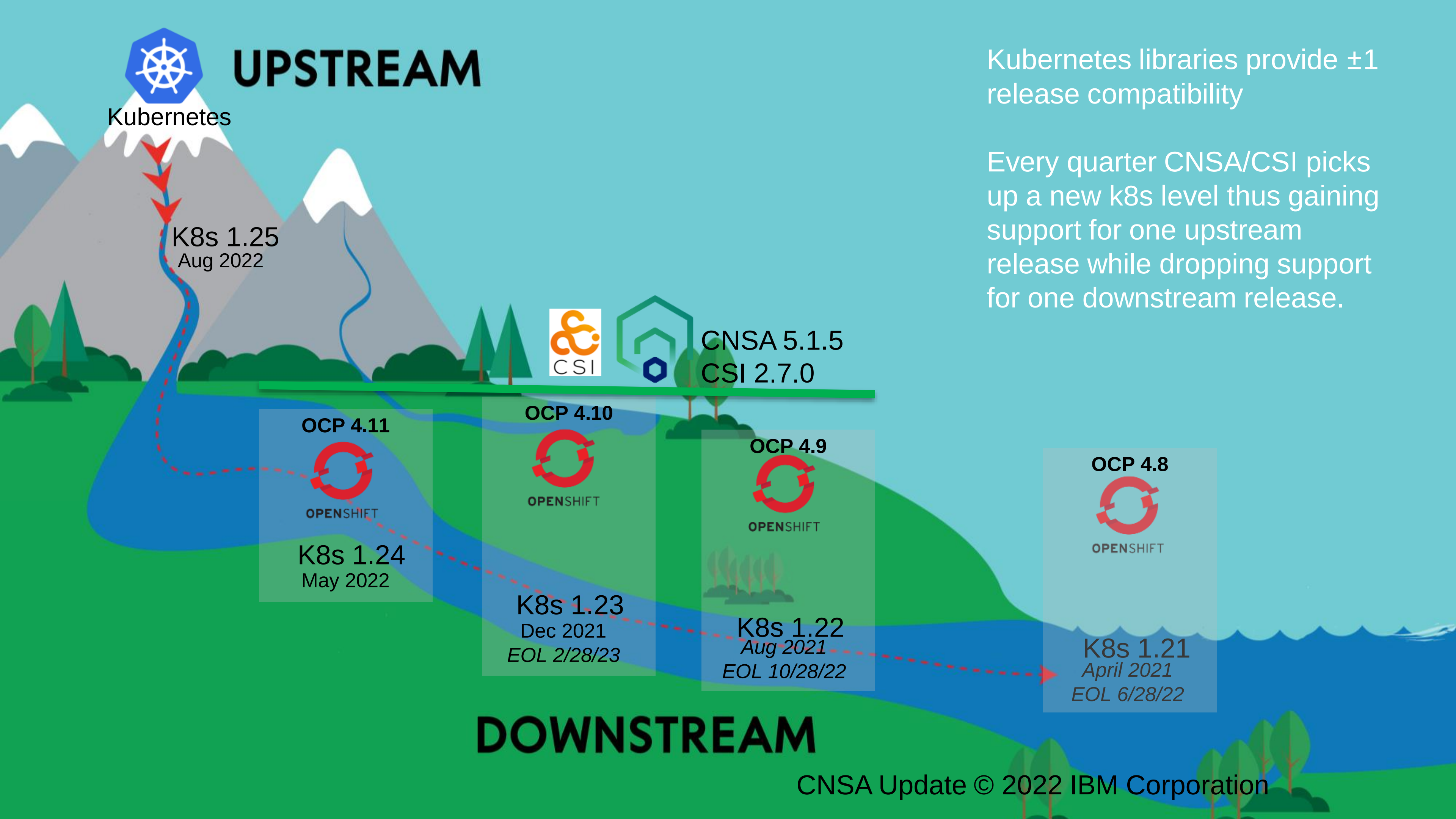


IBM Spectrum Scale & containers
(past / present / future)

**Roadmap indicates intention not a statement of commit*

Spectrum Fusion
Spectrum Scale as global data plane
Spectrum Protect as data protection layer





Kubernetes libraries provide ±1 release compatibility

Every quarter CNSA/CSI picks up a new k8s level thus gaining support for one upstream release while dropping support for one downstream release.

5.1.3.1

- Spectrum Scale 5.1.3.1
- CSI 2.5.1
- OCP 4.10
- Must-gather image moved to IBM container registry as public image
- Storage Class v2
- Consistency Groups
- Compression of PVs
- Storage Tiering of PVs

5.1.4

- Spectrum Scale 5.1.4
- CSI 2.6.0
- fsGroup on RWO
- statefulSet-> deployment

5.1.5

- Spectrum Scale 5.1.5
- CSI 2.7.0
- OCP 4.11
- Pod Disruption Budgets to moderate updates driven bei Machine Config Operator (MCO)
- Application awareness on upgrades and config changes
- Fixed issue of parallel reboots on upgrades and config changes
- CoreDNS service replaces /etc/hosts files
- Admin and deamon names in Scale cluster migrated to FQDN
- UpgradeApproval resource
- Support of moving daemon network onto CNI
- Reduced footprint of CNSA pods

Upgrade/config change flow **prior 5.1.5**

(in remote mount config)

- Config change made that requires core pod restart or new code level deployed
- Operator figures it needs to restart pods
- Operator deletes core pods
  - Quorum is protected, pods with quorum designation are restarted one-by-one and only if this will not result in quorum loss
  - Non-quorum pods will be restarted in parallel
- If kernel module failed to unload, worker node is rebooted

Issues **prior 5.1.5**

- When core pods go down the file system is pulled away under active application pods resulting in I/O errors. Application pods will therefore just fail instead of shutting down gracefully
- It takes k8s a few minutes to notice that the application pod failed and need to be rescheduled.
- Applications might be exposed with no replica available to take over. In this case an application outage will be the result of the core pod restart.
- Restarting all non-quorum pods at once will most likely bring whole replicaSets down again resulting in application outage.
- Having the application pods still around makes it more likely that kernel module unload fails

Upgrade/config change flow **in 5.1.5**

(in remote mount config)

- Config change made that requires core pod restart or new code level deployed
- Operator figures it needs to restart pods and rolls cluster one-by-one
- Operator cordons node
- Operator drains node
  - Application pods shut down gracefully and are rescheduled on another node
  - Pod disruption budgets are honored
- When all application pods got deleted the CNSA core pod gets deleted as well
- If necessary node will be rebooted, otherwise just core pod is restarted
- Node gets uncordoned

Consequences

- Instead of failing the application pods they are shut down and rescheduled. File system stays available as long as application pods are up
- Rolling one-by-one has less impact on application availability but results in update to take more time
- Update might get stuck if drain fails, for example because it is blocked by a pod disruption budge
- A full drain also affects pods that do not consume PVs from CNSA.
- Having all consumers of the file system shut down makes it more likely that the kernel module can be unloaded and a reboot is avoided

Machine Config Operator (MCO) config change flow **prior 5.1.5**

- Config change made that requires MCO to reboot node
- MCO cordons node
- MCO drains node
  - This results in deletion of CNSA core pod
- When all pods got deleted, MCO will reboot the node

Issues **prior 5.1.5**

- MCO has no understanding of Scale quorum and might unintentionally take down whole Scale cluster
- Deleting the CNSA core pod races with application pods. Some applications will loose file system acces before they are shut down
- On Fusion there is RAID tolerance and rebuilds to take into account as well.

Machine Config Operator (MCO) config change flow **in 5.1.5**

- Config change made that requires MCO to reboot node
- MCO cordons node
- MCO drains node
  - The pod disruption budget protects the CNSA core pod from deletion
  - The CNSA operation will be notified of the drain and checks if it is safe to restart the node (from a Scale perspective)
  - When it is safe and application pods got deleted the operator will patch the pod disruption budget to allow deletion of core pod
  - When all pods got deleted, MCO will reboot the node

Consequences

- MCO will not be able to bring down a node that is vital for the Scale cluster to stay up
- The CNSA core pods will stay up until all application pods are drained. Therefore the file system will stay available as well.
- If we are in a situation where bringing down a core pods is unsafe the reboot will be blocked. For example, if one of three quorum nodes is already down, bringing down another one is blocked.
- This might result in a deadlock in case the reason for blocking the reboot is persistent.

UpgradeApproval

- In order to fully complete a Spectrum Scale upgrade the clusterMinReleaseLevel needs to be set to the new version
- Once this is done, the upgrade becomes irreversible
- After the now images have been rolled out to the worker nodes the operator creates a new resources called `upgradeApproval`
- In order to approve, just patch this resource and the operator will increase the clusterMinReleaseLevel
- For remote mount you may need to increase fsVersion on storage cluster (on Fusion we have an `upgradeApproval` for this as well)

CoreDNS

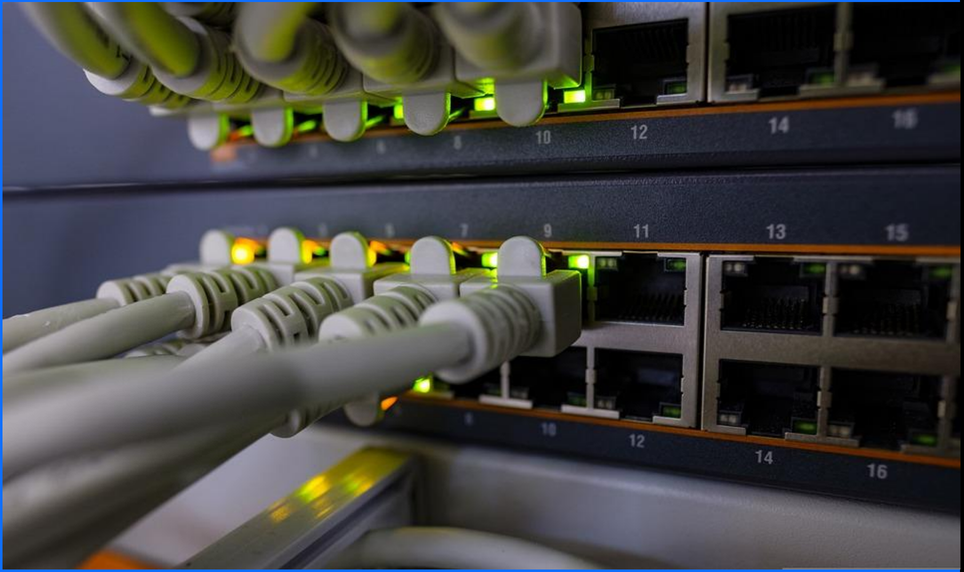
- Prior to 5.1.5 CNSA injected entries into core pod's `/etc/hosts` file to help with name resolution of Scale admin&daemon networks
- The drawback of this solution was that any change on `/etc/hosts` required a pod restart. For example, just adding a node resulted in restarting all core pods in the cluster. This was amplified with the non-graceful way we did restarts prio to 5.1.5
- With 5.1.5 we replaced `/etc/hosts` by our own coreDNS service that can be dynamically updated. It runs in its own namespace: `ibm-spectrum-scale-dns`
- As a result nodes can be added without impact on running applications

Fully Qualified Domain Names

- In 5.1.5 we change the node names for admin and daemon network to fully qualified names. This helps with intercluster features like metroDR, regionalDR, data fabric, etc.
- This means during 5.1.5 upgrade the previous shortnames get migrated to FQDN. From a Scale perspective we are renaming every node in the cluster.
- The new names are structured like this:

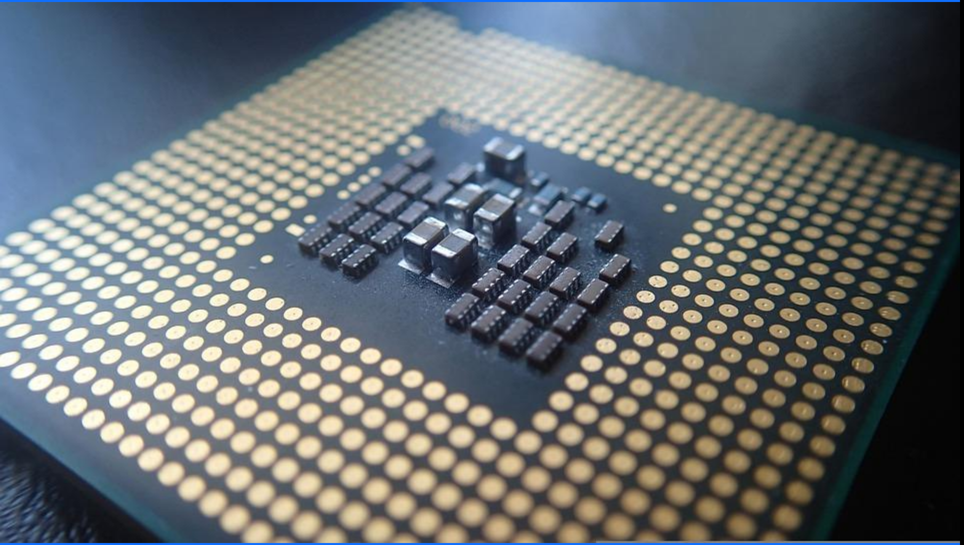
```
<node-shortname>.<admin|daemon>
.<namespace>.stg.<basedomain>.
```

CNI



- By default CNSA uses the host network
- This means both admin and daemon network run on the worker node IP.
- This does not require any configuration up-front. But it has some disadvantages
  - Potential port conflicts with other daemons running on the host
  - Full exposure on host network, also host network fully exposed to core pod. Not optimal from a security point of view
  - No choice of different HBA
- With 5.1.5 we support additional networks defined via Container Network Interface
  - Up-front configuration of CNI network needed
  - Admin network will use standard pod network
  - Daemon network will run on CNI
  - Network policies are supported on certain CNI types
  - No exposure on host or potential port conflicts

Resources



- Previously the resource requirements for our pods where not properly documented. Many people confused the minimum hardware requirement with our actual resource consumption
- Documentation has now been updated to reflect our resource consumption
- CNSA core pods request by default 25% of worker node resources. This fits nicely for Fusion (with ECE), but is a bit on the high side for remote mount setups.
- CNSA core pod requests can be dialed down to 1000mCPU and 2GiB. Allocating more might result in better performance. CPU might be dialed down even futher, but on your own risk.
- CNSA core pods set memory limits very high. This prevents us from being killed by kubelet for using too much memory. However the pods should never take significantly more than requested.
- CNSA core pods set CPU limits very high. This enables us to use free CPU cycles. On load we get throttled down to the request.

Auto configuration of tunables related to memory footprint

CNSA calculates optimal values for several memory related tunables from the request specified in the Cluster CR

The concept behind this is that the user tells us how much memory we can use, and the code figures out how to use it best.

request	mftc/msc	worker_threads	mftc_msc_mem	token_memory	pagepool	comment
2 GiB	16k	128	160 MiB	58 MiB	0.79 GiB	clients only
4 GiB	48k	256	480 MiB	174 MiB	2.36 GiB	clients only
8 GiB	96k	1024	960 MiB	348 MiB	4.72 GiB	
16 GiB	224k	1024	2240 MiB	812 MiB	11.02 GiB	
32 GiB	256k	1024	2560 MiB	928 MiB	26.59 GiB	
64 GiB	256k	1024	2560 MiB	928 MiB	58.59 GiB	
128 GiB	256k	1024	2560 MiB	928 MiB	122.59 GiB	

Consistency Groups

Fileset Scaling

- CSI 2.5.0 introduced **`storageClass`** 2.0 which changed the mapping of PVs to file system structures
- The new **`storageClass`** maps
  - **`namespaces`** to independent filesets.
  - **`persistentVolumes`** to dependent filesets inside the indepentent ones of the enclosing namespace
- In parallel we worked on raising the limits for filesets
  - Up to 3,000 independent filesets
  - Up to 50,000 dependent filesets (in 5.1.6)
  - No file system freeze on fileset snapshots
- The independent fileset also acts as a consistency group allowing to create a single snapshot across all PVs in a namespace.

Down the road

Todd, the admin
stuck in implementation

What is a gateway node and why would I want one?

How many NSD servers do I need?

When I expand the fs is it ok to create just one NSD with the additional capacity?

Should I have dedicated metadata disks?

How do I choose quorum nodes?

Todd, the admin

met on his level of abstraction

Dallas is our async DR site

The apps work with data from our S3 store

This project has about 50TiB of data now, but is expected to grow to 250TiB

Our satellite site needs to process data from our headquarters

We have a two-site sync mirror setup for HA

Example:
Cloud Filesystem

The admin only requests capacity per tier

The operator takes care about all the details: How many NSDs to provision, which designation (data/metadata), where to attach, replication, etc.

In future we want to be able to have a file system that inside boundaries grows automatically as more and more data gets stored.

```
apiVersion: scale.spectrum.ibm.com/v1beta1
kind: Filesystem
metadata:
  labels:
    app.kubernetes.io/instance: ibm-spectrum-scale
    app.kubernetes.io/name: cluster
  name: cloudcsi-sample
  namespace: ibm-spectrum-scale
spec:
  local:
    pools:
    - name: system
      capacity: 1Ti
      storageClass: ibm-spectrum-scale-managed-io2
    - name: data
      capacity: 2Ti
      storageClass: ibm-spectrum-scale-managed-gp2
```

What’s in the pipeline?

Thank You!