Pending RPC Detection & Expel

(aka preventing a single bad node from bringing down your cluster)

Spectrum Scale German User Meeting 2023
Sindelfingen, Germany – March 22-23, 2023

Mathias Dietz (IBM) – Spectrum Scale RAS Architect
mdietz@de.ibm.com

Disclaimer

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.
- IBM reserves the right to change product specifications and offerings at any time without notice. This publication could include technical inaccuracies or typographical errors. References herein to IBM products and services do not imply that IBM intends to make them available in all countries.

Cluster-wide impact due to single misbehaving node

Spectrum Scale is a highly distributed system which delegates certain operations to individual nodes in order to achieve high performance.

Some operations are limited to nodes with specific node designations, while some can be fullfilled by any cluster node. Examples are:

- Token Management
- Byte range lock Management
- Inode Allocation
- Block Allocations
- Metadata updates (metanode)
- etc.

Plain client nodes (even remote clients) can play an important role in the cluster (e.g. metanode for a file, token owner,..) and therefore it is essential for the cluster health to identify and expel misbehaving nodes.

Expels – Why do We Need Them and What is Their Impact?

What is an expel?

An expel refers the case in which a node is temporarily suspended from accessing a Spectrum Scale cluster's file systems. An expelled node is sometimes referred to as having "lost cluster membership" or "lost quorum" (note this is different use of the term "quorum" than that used in "cluster node quorum" or "file system descriptor quorum").

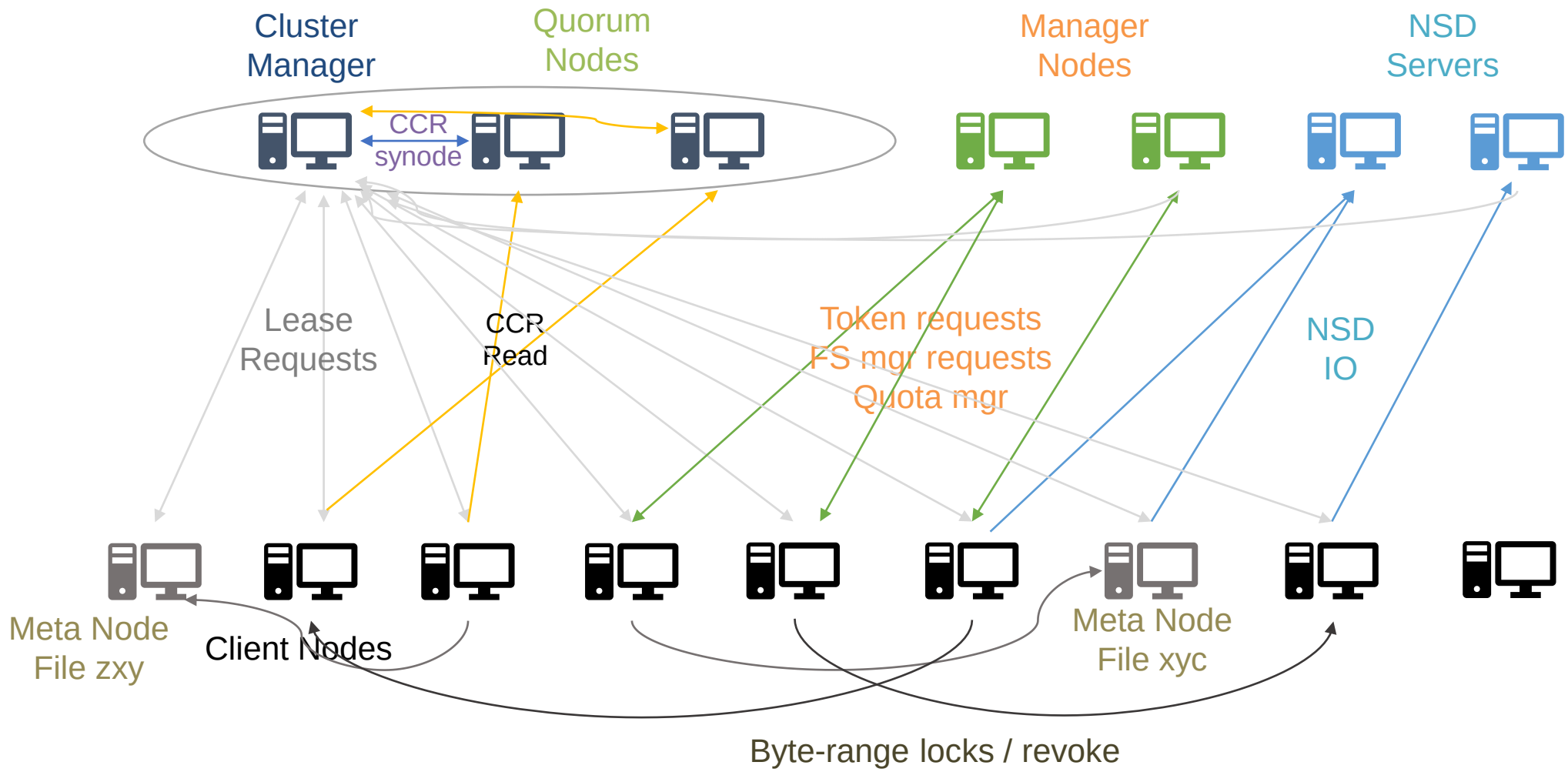
Impact

After a node is expelled from a cluster, the expelled node may no longer submit I/O requests, its tokens and locks are released, and its Spectrum Scale file system(s) are unmounted. Applications executing on the expelled node may fail after receiving an EIO (input/output error) or ENOENT (no such file or directory) response to an I/O request made to an unmounted file system. After an unmount occurs, an application may receive a terminating SIGBUS signal if its binary is stored on the file system that gets unmounted.

Why Does Spectrum Scale Need to Expel Nodes?

To ensure data integrity and good performance, while preventing hangs and deadlocks, Spectrum Scale requires reliable communication between all nodes using a file system

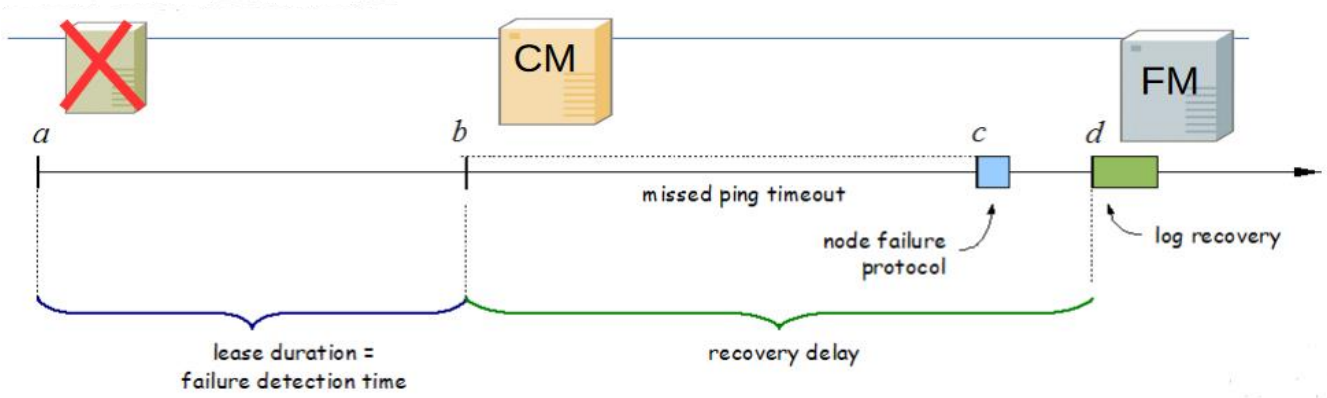
Spectrum Scale Network Communication



What happens if a single node behaves badly ?

There are multiple different failure scenarios which need to be considered:

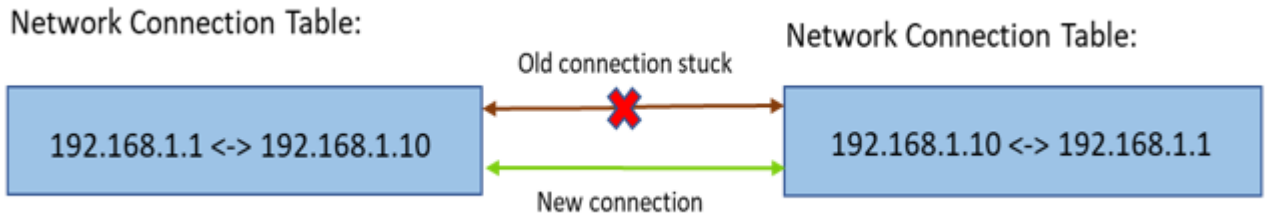
1. Full Node Outage (e.g. power loss) or Network unreachable
   - Disk lease expires -> Cluster Manager expels node from cluster



2. Network Connectivity between two nodes
   - Node A cannot talk to node B but both can talk to the Cluster Manager, then CM will decide which node to expel (A or B):
     - quorum nodes over non-quorum nodes
     - local nodes over remote nodes
     - manager-capable nodes over non-manager-capable nodes
     - nodes managing more FSs over nodes managing fewer FSs
     - NSD server over non-NSD server
     - Otherwise, expel whoever joined the cluster more recently
     - A custom callout script can be configured to customize the behavior

What happens if a single node behaves badly ?

3. Instable network / dropped connections
   - Proactive reconnect the network connection . Eventually expel node if reconnect fails

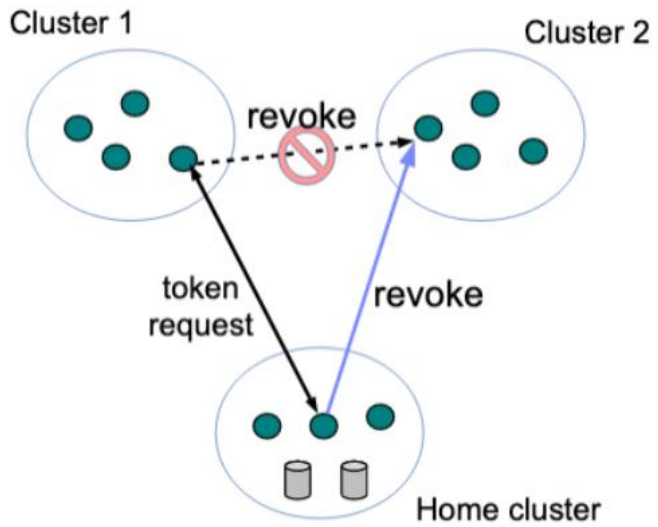


4. Stuck in IO (aka IO Hang)
   - IO Hang detection.
     Recommended option: panicOnIoHang=yes (default on ESS)

5. Blockage of critical threads and deadlocks
   - Critical thread monitoring, Automatic Deadlock detection should be enabled

What happens if a single node behaves badly ?

6. Partially unresponsive node
   - Continues to respond to disk lease renewals and some rpcs -> No expel
   - BUT: does not respond to more heavyweight RPCs like token revokes
   - Often seen due to CPU or memory starvation (OOM) on the node
   - Solution: Pending RPC detection and expel



Picture: token request / revoke in a cross cluster setup

Health Monitoring of pending RPCs and automatic expel

Starting with 5.1.6 mmhealth has the functionality to detect pending RPCs for token revokes

- When then RPC age is exceeds a given warning threshold it will raise a rpc_waiters event to inform the user about the misbehaving node
- Optionally a expel threshold can be configured when mmhealth should start to expel the node automatically

- There are two mmchconfig options to control the feature:
  - mmhealthPendingRPCWarningThreshold (default = 180 seconds)
  - mmhealthPendingRPCExpelThreshold (default = 0 means disabled)

Detection of pending RPCs - rpc_waiters warning event

The mmhealth GPFS monitor runs "mmdiag --network -Y" every 2nd monitoring cycle (default interval for GPFS monitoring is 30s, means that rpc monitoring is done every 60s).

```
mmdiag:pendingMessages:HEADER:version:reserved:reserved:messageId:serviceNumber:version:typeNumber:typeString:ageInSeconds:sentToNumberOfNodes:waitingForReplyFromNumberOfNodes:this:xhold:resending:cl:cbFn:sentByThreadId:sentByThreadName:sentBySenderAddress:replies:destination:node:status:error:replyLength:RDMAConnectionIndex:TCPConnectionIndex:isResending:
    mmdiag:pendingMessages:0:1:::1881:::18:tmMsgRevoke:305:1:1:::::::2427972:SharedHashTabFetchHandlerThread::10.0.100.120:<c0n1>:pending:0:0:0::
    mmdiag:pendingMessages:0:1:::1881:::18:tmMsgRevoke:108:1:1:::::::2427972:SharedHashTabFetchHandlerThread::10.0.100.88:<c1n1>:pending:0:0:0::
    mmdiag:pendingMessages:0:1:::1881:::18:tmMsgRevoke:508:1:1:::::::2427972:SharedHashTabFetchHandlerThread::10.0.100.74:<c1n0>:pending:0:0:0::
    mmdiag:pendingMessages:0:1:::1881:::18:tmMsgBRRevoke:400:1:1:::::::2427972:SharedHashTabFetchHandlerThread::10.0.100.70:<c1n3>:pending:0:0:0::
    mmdiag:pendingMessages:0:1:::1895:::3:commMsgCheckMessages:4:1:1:::::::2423979:EEWatchDogThread::10.0.100.120:<c0n2>:pending:0:0:1::
.....
```

If the mmdiag output contains some pending RPCs with an age larger than the mmhealthPendingRPCWarningThreshold , it will raise a ***rpc_waiters*** event which will change the GPFS component state to DEGRADED.

- With 5.1.6.0 only tmMsgRevoke RPCs are detected
- With 5.1.6.1 tmMsgRevoke and tmMsgBRRevoke RPCs are detected

Multiple pending RPCs will be grouped in a single rpc_waiters event. If mmhealthPendingRPCWarningThreshold is set to 0, no event will be created.

Automatic expel of unresponsive node

Enable automatic expel by setting a threshold in seconds using mmchconfig. The expel threshold should be higher than the warning threshold

e.g. mmchconfig mmhealthPendingRPCExpelThreshold=300

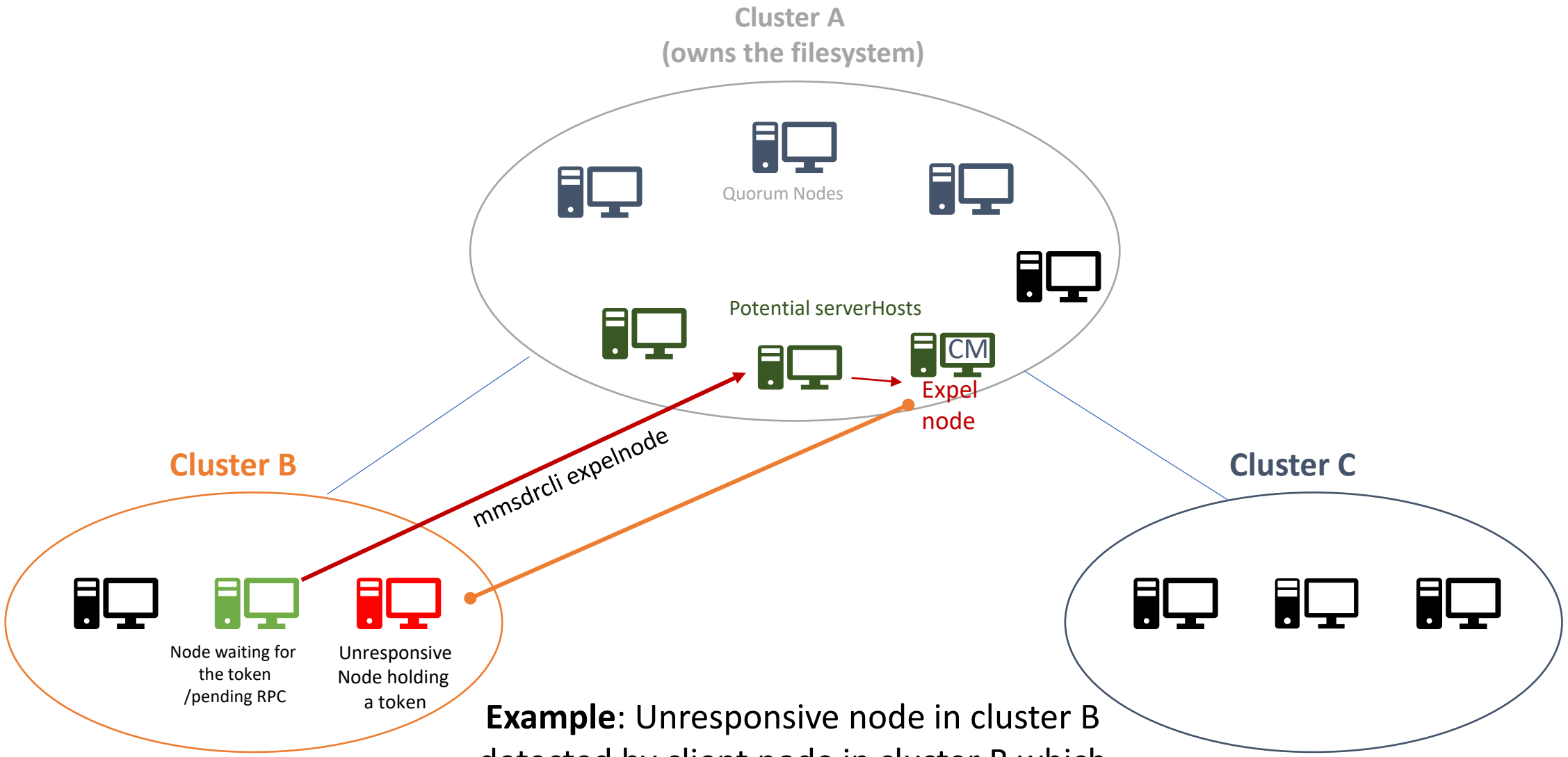
if mmhealthPendingRPCExpelThreshold is set to 0 (default), no expel will be done

If the mmdiag output contains some pending RPCs with an age larger than the mmhealthPendingRPCExpelThreshold, it will:

- get the node to expel (victim) from the pending rpc output
- get the node (serverHost) where we send the expel request to from the connected node list, but matching the cluster id (e.g. <c1n3>)
- run *mmsdrcli expelnode* command with the victim and serverHost as parameters
- raise a ***rpc_waiters_expel*** warning event (non-state-change) for the particular node expel

- By default the node will be expelled forever (no-rejoin). Once a node is expelled, the node will not rejoin the cluster unless the customer has run *mmexpelnode -r -N <node>* on the home cluster

Spectrum Scale – Multicluster Expel



Example: Unresponsive node in cluster B detected by client node in cluster B which will trigger expel from cluster A

Health Monitoring of pending RPCs and automatic expel

Spectrum Scale 5.1.6.0

- Added mmhealth support for pending RPC detection and automatic expel
  - All nodes in the cluster (s) must run >= 5.1.6.0
  - If there are nodes in the cluster with an older level:
    - Pending RPCs on those nodes will not be detected and expels send to those nodes fail silently
- Only tmMsgRevoke RPCs are detected

Spectrum Scale 5.1.6.1

- Detection of tmMsgBRRevoke RPCs added

Spectrum Scale 5.1.7.0 improvements

- Retry sending expel requests to other nodes in case of failure

Spectrum Scale 5.1.7.1

- Prior to 5.1.7.1 some RPCs did not update the age correctly (age of 0s) and would not be detected

If you are running an older version of Scale and can not upgrade -> request an eFix

findSlowNodes

- findSlowNodes (originally delivered as findSlowMinions) is a script developed by our development team as a temporary solution given to some customers to address the problem of nodes not responding to tmMsgRevoke RPCs, which lead to cluster hangs
- This temporary solution was intended to allow customers to find such unresponsive nodes and offer guidance on how to expel such nodes
- Some customers have choosen to use this temporary (stop-gap) solution, instead of the official mmhealth and GPFS core solution, because of challenges related to upgrading to the official solution.
- We recommend moving to an official solution, and Efixes can be requested through our support team, because there are some disavantages to using find slowNodes:
  - Lacks full automation for expelling problematic nodes
  - Does not integrate into mmhealth’s monitoring/alerting ecosystem.
  - Not tested by our test teams.

Thank you for using
IBM Spectrum Scale!