**pureScale with Pacemaker - Chapter 2: Have we reached quorum?**

A diagram of a network

AI-generated content may be incorrect.

Welcome back to the pureScale with Pacemaker blog series! If you have not already, I recommend visiting the first blog post in this series, '[pureScale with Pacemaker - Chapter 1: The Big Picture](https://www.idug.org/news/purescale-with-pacemaker---chapter-1-the-big-picture)', to get an in-depth understanding of the pureScale resource model with Pacemaker. In this second blog entry, we explore the topic of cluster quorum, and how Pacemaker determines the health state of each host and the cluster as a whole. But first, let’s picture the textbook “*split-brain”* scenario.

Imagine a network failure occurs within the cluster, causing some hosts to lose communication with each other. In this situation, if all nodes are allowed to continue to perform I/O on the same set of shared data, we may encounter data corruption! To prevent this situation, Pacemaker must answer many questions during any software/hardware failure:

* What groups of hosts are still able to communicate with each other?
* Has communication been established between enough hosts to allow normal cluster operations to proceed?
* Which hosts should stay operational and which should be shut down?
* Should the cluster shut down completely due to lack of communication/online hosts?

All these questions are answered through the concept of quorum!

**What is quorum?**

*Quorum* is the mechanism that determines if a host is permitted to be an active participant within the pureScale cluster. As an active participant, a host is permitted to manage cluster resources and perform HA related tasks such as executing recovery actions when the cluster experiences software or hardware failures.

When determining quorum on a host, Pacemaker queries how many cluster hosts are online and can be communicated with from the current host. To achieve quorum, a host must be able to communicate with a minimum number of other cluster hosts.

How does Pacemaker know if other cluster hosts can communicate with the current host? During the quorum decision process, Pacemaker counts the number of “quorum votes” the host has received. With the new Pacemaker cluster manager, Db2 adopts a “vote” quorum policy where each cluster host provides one quorum vote. A host is only able to provide its vote to another host’s quorum decision if it is online and able to communicate with the other host.

What happens if a host cannot reach quorum? If a host cannot communicate with a minimum number of other hosts in the cluster, it will be brought offline through fencing. This process will be discussed in more detail in the coming sections.

As a general rule, the pureScale cluster can only remain operational if a minimum number of hosts have reached quorum, consequently allowing them to remain online and perform cluster operations. Note that this “minimum number” is the same minimum number needed for an individual host to reach quorum. Therefore, we can define *cluster quorum* as the minimum number of hosts that must be active and communicative for the cluster to stay online.

**Types of Quorum Policies**

For pureScale deployments, Pacemaker supports three different methods for determining the minimum number of quorum votes needed for a host to reach quorum. These are known as *quorum policies*. Most quorum policies can only be used for certain cluster configurations, taking the decision out of which quorum policy should be used. We’ll go through each quorum policy in the subsequent sections.

**Two-node Quorum**

Two-node quorum is the quorum policy used for a cluster that contains only two hosts. In this scenario, the cluster will be in a collocated member/CF scenario where each host houses a member and a cluster caching facility (a.k.a CF). If the hosts lose communication with each other, the host that runs as the Storage Scale cluster manager will stay online, consequently bringing the other host down.

If another host is added to the cluster, bringing the total number of hosts in the cluster to 3 or more, the quorum policy will automatically be updated to *majority quorum*.

**Majority Quorum**

Majority quorum is the default quorum policy for a pureScale cluster that contains more than two hosts. For a host or cluster to maintain majority quorum, a minimum of half the cluster hosts + 1 must be online and communication must be established between them. For example, in a cluster with 5 hosts (2 CFs and 3 members), quorum is maintained if at least 3 of the hosts are online and communicative.

This quorum policy is intended for clusters that contain an odd number of hosts. What happens if we have an even number of hosts? An alert will be generated to inform the user that their cluster may not remain operational in certain failure scenarios. One scenario includes a network error that causes an equal split of quorum votes. In this scenario, we end up with two “partitions” of hosts where each partition contains an equal number of hosts that can only communicate with each other. In this case, quorum cannot be reached on either side and a full cluster outage will occur. The next quorum policy offers a solution to this scenario.

**Majority Quorum with a Quorum Device**

This quorum policy follows the same principles discussed previously for majority quorum, with the addition of a quorum device.

A *quorum device*, commonly known as a *qdevice*, is a third-party arbitrator that provides an “extra” quorum vote to the cluster quorum decision. Only one quorum device can be configured per pureScale cluster. With this quorum policy, quorum in the cluster is maintained if at least half of the cluster hosts and the quorum device are online and communicative.

One advantage of a quorum device is the ability for a cluster to sustain more host failures than standard majority quorum would allow. For example, say we have a cluster containing 4 hosts (2 CFs and 2 members). This cluster requires at least 3 hosts to be online and communicative. Any software/hardware that causes half the hosts to shutdown will result in a total cluster outage. Now with a quorum device configured, the total number of cluster quorum votes has been increased to 5. We still require at least 3 quorum votes to maintain quorum, but one of these quorum votes can now be provided by the quorum device. This allows the cluster to stay operational even when half of the cluster nodes are unreachable/offline!

Another advantage of a quorum device is the ability for the cluster to remain online during an even cluster split. As illustrated in the “Majority Quorum” section above, both sides would be shut down if a quorum device was not configured. Now with a quorum device, one partition of hosts will win the quorum device tie-breaker vote, resulting in this partition’s hosts remaining online. The losing partition’s hosts will be forced to shut down.

**When should a quorum device be used?**

A quorum device should only be required when a cluster contains an even number of hosts. Illustrated in the example above, the total number of cluster quorum votes was increased by 1, creating an odd total number of available cluster quorum votes. With an odd number of votes, there is no possibility of a tie between two partitions of hosts proposing different cluster recovery options. In the case where an equal number of hosts supports two different recovery actions, the quorum device provides one final quorum vote to break the tie. The quorum device will provide its vote to the partition of hosts that contains the lowest node ID.

Why shouldn’t a quorum device be used in a cluster with an odd number of hosts? This restriction is most easily explained through another example. If we have a cluster that contains 5 hosts, the total number of cluster quorum votes is 5 and quorum is maintained if we have at least 3 hosts online. Now if we add a quorum device, the total number of cluster quorum votes is increased to 6, bringing majority to a minimum of 4 votes. In this scenario, adding the quorum device does not enhance overall cluster availability, and requires the quorum device host to be kept online at all times.

**Requirements for a quorum device**

The host that runs the quorum device does not need to have the same characteristics of the other hosts in the pureScale cluster, as its sole purpose (from pureScale’s point of view) is to run the quorum device. In other words, this host does not require the same amount of system resources (ex. CPU, RAM, storage) as a regular Db2 node. The specific requirements for the quorum device host can be found in the [Prerequisites for an integrated solution using Pacemaker](https://www.ibm.com/docs/en/db2/12.1?topic=pacemaker-prerequisites-integrated-solution-using) documentation.

Similar to regular cluster hosts, the host must be accessible to all other hosts in the cluster configuration in order to provide its quorum vote. A technical difference resides in the protocols used for communication. The quorum device host uses TCP to communicate with the cluster hosts over port 5403, while the cluster hosts communicate using UDP on 5404.

Unlike a regular cluster host, the quorum device host must not be a part of the pureScale cluster configuration and Db2 does not need to be installed. Since the quorum device host is not tied to a specific cluster, this host can act as a quorum device for multiple Db2 instances at the same time! A quorum device being used by a pureScale cluster can also be the same quorum device used by another pureScale, HADR, MF, or DPF cluster. The only restriction with multiple instances using the same quorum device is that the Pacemaker domain name for each cluster must be unique.

**How do I set up a quorum device?**

Setting up a quorum device is fast and easy! Firstly, the *corosync-qdevice* packages must be installed on all cluster hosts and the *corosync-qnetd* packages must be installed on the quorum device host. During pureScale feature installing, the *corosync-qdevice* packages should be installed on cluster hosts by default.

Once all packages are installed, passwordless root SSH and SCP must be temporarily enabled between all cluster hosts and the quorum device host. Note that this authentication method and file transfer protocol is only required during the set up or removal of a quorum device or when performing further cluster configuration changes to a cluster that contains a quorum device. These configuration changes include deleting a pureScale instance using the *db2idrop* command or adding/dropping a member or CF using the *db2iupdt* command.

Once all requirements are met, you can simply run the *db2cm -create -qdevice <hostname>* command to create the quorum device. That’s it! For more information about configuring a quorum device, visit the documentation [Installing and configuring a quorum device.](https://www.ibm.com/docs/en/db2/12.1?topic=SSEPGG_12.1.0/com.ibm.db2.luw.admin.ha.doc/doc/t_install_config_quroum_device.htm)

**How can I check the quorum policy used by my cluster?**

Now that we’ve gone through all the possible quorum policies that can be used within a pureScale cluster, I’m sure you are wondering how to determine which policy is employed on your configuration.

The *db2cm* utility can be used to display quorum information and monitor the quorum device connectivity status (if one exists). To view quorum information on each host, *db2cm -list -quorum* can be run. This command will display the following information:

* Quorum policy used in the cluster labeled as “Quorum type”
* Total number of quorum votes received by current host as “Total Votes”
* The minimum number of quorum votes needed for the host/cluster to remain online labeled as “Quorum Votes”
* List of all the hosts within the cluster that provide a quorum vote labeled “Quorum Nodes”

What can we expect the output to look like? The following examples show a sample output of *db2cm -list -quorum* for each quorum policy:

**Output of Two-node Quorum**A black screen with white text

AI-generated content may be incorrect.

As expected, the quorum information listed for a two-node quorum policy shows a total of two quorum votes for the cluster. The “Quorum Votes” section indicates that only one host needs to be online for the current host and cluster to remain operational.

Notice that the quorum type displayed contains the words "wait for all disabled". This policy allows Pacemaker to start any one host in the cluster without requiring the other host to be online at the same time.

**Output of Majority Quorum**

A black screen with white text

AI-generated content may be incorrect.

In the above output, we see that the cluster contains a total of 4 quorum votes, directly correlating to the number of hosts lists under the “Quorum Nodes” section. For quorum to be reached, three quorum votes must be available. In other words, at least three hosts must be online and only one node failure can be sustained before a total cluster outage occurs. Note that this cluster has an even number of nodes and ideally should be employing a quorum device.

**Output of Majority Quorum with a Quorum DeviceA black screen with white text

AI-generated content may be incorrect.**

Lastly, we see the same cluster from the last example, but now with a quorum device configured. Note the change in quorum policy from “Majority” to “Qdevice”. Another difference from the previous example is the increased total number of quorum votes. The cluster still requires three votes to maintain quorum, but the cluster can now sustain up to two node failures before the cluster is brought offline. One last thing to be aware of is the absence of the quorum device host name under the “Quorum Nodes” section. Since the quorum device is not a dedicated host within the cluster configuration, it will not be listed under this section.

To see the host name of the quorum device and its connectivity status to the cluster, you can run the *db2cm -status* command as the root user. Sample output of this command where the quorum device runs on the host *qdevice-host* is shown below:

If a quorum device is not configured, the output of this command will display the text “Not configured”.

**Fencing**

The last topic we will touch upon is fencing. During a software/hardware failure scenario, fencing is the mechanism used to block the failed host from accessing cluster-shared resources. The Db2 pureScale resource model uses a fence resource, controlled by the db2fence\_ps resource agent, to block fenced hosts from accessing the cluster filesystem and communicating with the rest of the cluster hosts. For more information regarding the fence resource, visit the first blog post in this series, '[pureScale with Pacemaker - Chapter 1: The Big Picture](https://www.idug.org/news/purescale-with-pacemaker---chapter-1-the-big-picture)'.

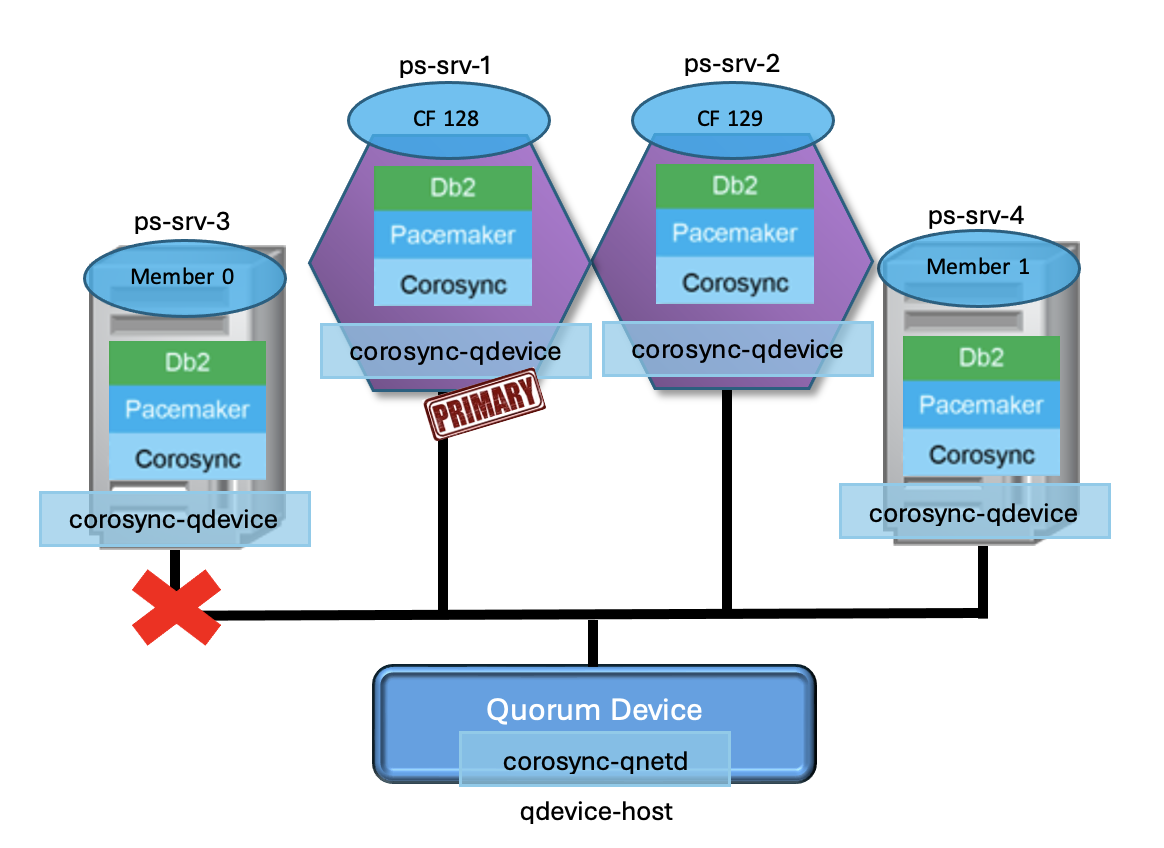
During a split-brain scenario, fencing works together with quorum to stop partitions of hosts from corrupting each other’s data. Once a failed host has come back online, recovery will be performed to unfence the host and allow it to rejoin the cluster.

**Network Failure Scenario**

Now with all the information you’ve learned about quorum and fencing, it’s time to look at a real failure scenario! Let’s revisit the first scenario we looked at, the network failure. In the image below, we have a four-host cluster configuration of two CFs (hosts *ps-srv-1* and *ps-srv-2*) and two members (hosts *ps-srv-3* and *ps-srv-4*) with a quorum device configured (host *qdevice-host*). In this cluster, a total of 5 quorum votes are available and each host can maintain quorum if they receive at least 3 votes.

A network error occurs on member 0 (host *ps-srv-3*), making communication to all other hosts in the cluster impossible.

To monitor the cluster and host’s quorum status, we will run commands and observe logs on the DC (i.e. Designated Controller) host and the failed host. The DC host can be determined from the output of the *db2cm -status* command, presented as “Current DC”. In our example, the DC host is *ps-srv-1*.



**During the failure**

Once the network failure has occurred on *ps-srv-3,* we have many tools to monitor the quorum and fencing status on each host. The pacemaker.log, found under the /var/log/pacemaker directory, documents all Pacemaker activity on the current host. Within this log, we first witness the DC host losing connection to the failed host.

|  |
| --- |
| Jan 01 00:03:08.503 ps-srv-1 pacemaker-attrd     [86226] (update\_peer\_state\_iter)   notice: Node ps-srv-3 state is now lost | nodeid=3 previous=member source=crm\_update\_peer\_proc  Jan 01 00:03:08.503 ps-srv-1 pacemaker-controld  [86228] (peer\_update\_callback)     info: Node ps-srv-3 is no longer a peer | DC=ps-srv-1 old=0x4000000 new=0000000  Jan 01 00:03:10.528 ps-srv-1 pacemaker-controld  [86228] (quorum\_notification\_cb)   info: Quorum retained | membership=125036 members=3 |

The logs taken from *ps-srv-1* show the current quorum status. Since we only lost connection to *ps-srv-3*, quorum is maintained on *ps-srv-1,* given that it has still received 4 votes. The “members=3" statement in the last log entry confirms that of the 4 cluster hosts, only 3 can be communicated with. The *db2cm -list -quorum* output also displays the decrease in total quorum votes *ps-srv-1* has received.

A black screen with white text

AI-generated content may be incorrect.

Returning back to our pacemaker.log on the DC node *(ps-srv-1),* we witness this host authorizing the fencing action to take place against the failed host. The pureScale fencing resource, *db2\_fence*, performs the fencing action. Once fencing has succeeded, the failed host has been declared offline and is no longer a member of the pureScale cluster.

|  |
| --- |
| Jan 01 00:03:11.567 ps-srv-1 pacemaker-schedulerd[86227] (pe\_fence\_node)    warning: Cluster node ps-srv-3 will be fenced: peer is no longer part of the cluster  Jan 01 00:03:11.575 ps-srv-1 pacemaker-schedulerd[86227] (schedule\_fencing)         warning: Scheduling node ps-srv-3 for fencing  Jan 01 00:03:11.596 ps-srv-1 pacemaker-fenced    [86224] (can\_fence\_host\_with\_device)       info: db2\_fence is eligible to fence (off) ps-srv-3: none  Jan 01 00:03:13.634 ps-srv-1 pacemaker-controld  [86228] (tengine\_stonith\_callback)         info: Fence operation 30 for ps-srv-3 succeeded  Jan 01 00:03:13.634 ps-srv-1 pacemaker-controld  [86228] (pcmk\_\_update\_peer\_expected)       info: crmd\_peer\_down: Node ps-srv-3[3] - expected state is now down (was member) |

The *db2cm -status* output on the DC node also shows the failed host has been declared offline. All host-specific resources are in the “Stopped” state and our resource for member 0 has failed over on to the other remaining member host (*ps-srv-4*).

*A screenshot of a computer

AI-generated content may be incorrect.*

Now what do we see from the failed host’s point of view? Looking in the failed host’s pacemaker.log, we first see the network error logged. Since the host has no connection to any other host, it is not able to obtain quorum. Without quorum, the failed host is no longer permitted to perform any sort of resource management or recovery operations on the cluster.

|  |
| --- |
| Jan 01 00:03:06  db2ethmonitor(db2\_ethmonitor\_db2inst1\_eth1)[23534]:    INFO: db2ethmonitor[23534]:644: /usr/lib/ocf/resource.d/db2/db2AdapterResponse.ksh ADAPTER\_OFFLINE ps-srv-3 0 eth1 db2inst1  Jan 01 00:03:13.417 ps-srv-3 pacemaker-controld  [6848] (abort\_transition\_graph)    info: Transition -1 aborted: Quorum lost | source=crm\_update\_quorum:447 complete=true  Jan 01 00:03:14.482 ps-srv-3 pacemaker-schedulerd[6847] (cluster\_status)    warning: Fencing and resource management disabled due to lack of quorum |

After these errors appear in the logs, we witness the failed host being fenced off successfully.

|  |
| --- |
| Jan 01 00:03:15.145 ps-srv-3 pacemaker-schedulerd[6847] (pe\_fence\_node)     warning: Cluster node ps-srv-3 will be fenced: db2\_member\_db2inst1\_0 failed there  Jan 01 00:03:15.322 ps-srv-3 pacemaker-controld  [6848] (tengine\_stonith\_callback)  info: Fence operation 4 for ps-srv-3 succeeded  Jan 01 00:03:15.322 ps-srv-3 pacemaker-controld  [6848] (pcmk\_\_update\_peer\_expected)        info: crmd\_peer\_down: Node ps-srv-3[3] - expected state is now down (was member) |

**Reintegration after the failure**

After the network issue on *ps-srv-3* has been resolved, the host will be unfenced and allowed to rejoin the cluster. In the pacemaker.log on the DC node, we see the following log entries illustrate that *ps-srv-3* has rejoinedthe pureScale cluster. It is also important to notice that quorum has now been retained with 4 members, corresponding to all 4 cluster hosts being able to communicate.

|  |
| --- |
| Jan 01 00:21:55.770 ps-srv-1 pacemaker-fenced    [86224] (update\_peer\_state\_iter)   notice: Node ps-srv-3 state is now member | nodeid=3 previous=unknown source=crm\_update\_peer\_proc  Jan 01 00:21:55.822 ps-srv-1 pacemaker-controld  [86228] (quorum\_notification\_cb)   info: Quorum retained | membership=125050 members=4 |

In the pacemaker.log on the originally failed host, we can also see that quorum has been restored.

|  |
| --- |
| Jan 01 00:22:26.336 ps-srv-3 pacemaker-controld  [6496] (pcmk\_\_corosync\_quorum\_connect)     notice: Quorum acquired |

Our *db2cm -quorum -list* output has also been updated to account for the newly recovered member host. Total quorum votes received to any cluster host now displays 5 (4 cluster host votes and the quorum device host vote).

Lastly, the *db2cm -status* output confirms all cluster hosts are online, member 0 has restarted on its original host, and the network resource for *ps-srv-3* is running.

A screenshot of a computer

AI-generated content may be incorrect.

**Wrap-up**

Our quorum discussion has now come to an end! Please reach out if you have any questions or concerns about this blog’s content. If you are interested in the quorum policies used in HADR and MF configurations, don’t miss '[The Book of Pacemaker - Chapter 4: Quorumania](https://www.idug.org/news/the-book-of-pacemaker---chapter-4-quorumania)' from 'The Book of Db2 Pacemaker' series. For more information about quorum policies used with Pacemaker on all HA configurations, visit the [Quorum policies supported on Pacemaker](https://www.ibm.com/docs/en/db2/12.1?topic=component-quorum-devices-support-pacemaker) documentation.

Want to learn even more about pureScale with Pacemaker? Keep an eye out for the next blog post in this series!