Proposal to make the failover of backend PostgreSQL nodes quorum aware.

Currently Pgpool-II proceeds to failover the backend node as soon as the backend health-check detects the problem or in case of an error occurred on the backend connection (when fail_over_on_backend_error is set).
This is good enough for the standalone Pgpool-II server, but when the Pgpool-II nodes are configured for high-availability and are part of the watchdog cluster, this technique of always performing a the backend failover whenever any Pgpool-II node detects the problem can cause problems like split-brain of PostgreSQL backends and/or can cause the data corruption(user ends up with multiple versions of data in different PostgreSQL servers).

The Problems and how this proposal is trying to solve those problems are discussed in detail latter in the document.

Purposed changes in the backend failover mechanism.

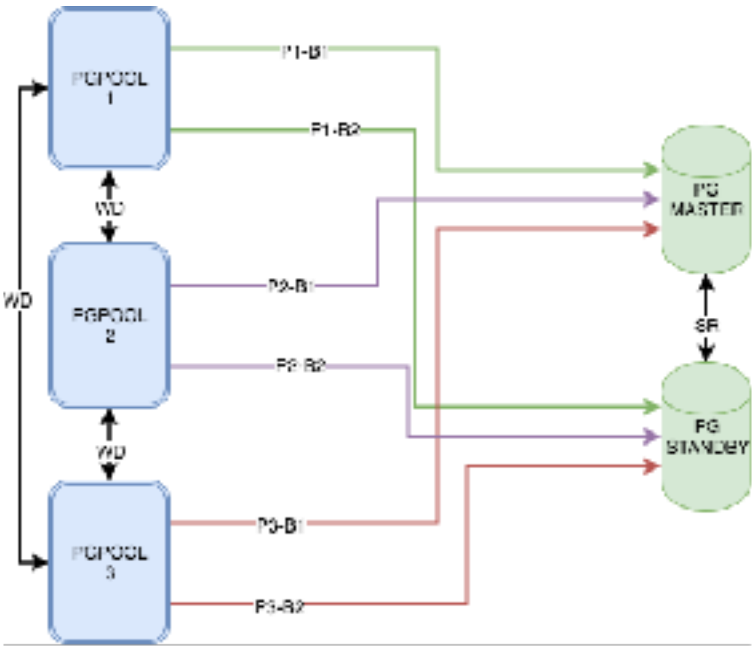
The proposal suggests to make the following changes in the backend failover mechanism when the Pgpool-II is configured with watchdog

- Pgpool-II should consult with all connected Pgpool-II nodes to confirm if the backend node is actually failed or not. and only perform the failover on the node when the majority of Pgpool-II nodes vote in favour of the failover.
- Pgpool-II should not promote the standby PostgreSQL server to Master PostgreSQL if the quorum is not present in the watchdog cluster.
- If the Pgpool-II detects that the standby PostgreSQL node is not reachable, but still it is reachable by majority of Pgpool-II nodes(failover disallowed by cluster majority). The Pgpool-II node should perform the local partial failover and mark the node as unreachable and as soon as the node become reachable again change its status back to “UP” without any user intervention.
- If a Pgpool-II node detects that the Master PostgreSQL node is not reachable, but is reachable by other Pgpool-II nodes(failover disallowed by cluster majority). The Pgpool-II node should disconnect all its active connection and keep disallowing the new connections until the connection issue with the master PostgreSQL is resolved.
- If the watchdog-master Pgpool-II node is having connection issue with backend while other Pgpool-II nodes in the cluster are disallowing the backend node failover for troublesome backend, The pgpool-II should resign from being the master watchdog node and let some other Pgpool-II node become the master watchdog node.

Situations where the current failover mechanism does not perform as it should.

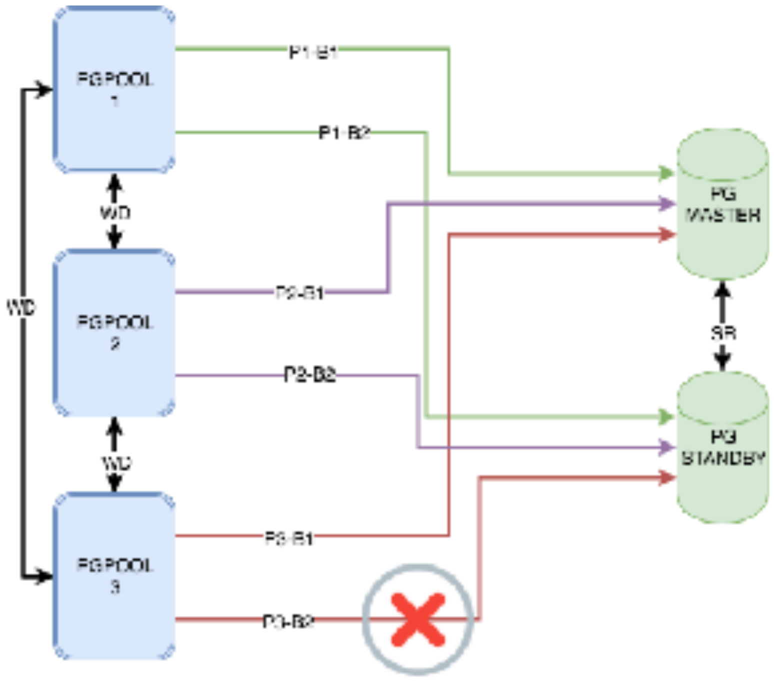
1. False Failover

Consider the simple and common high availability design, Where we have three Pgpool-II nodes (minimum required for the quorum settings) connected together through watchdog and two PostgreSQL streaming-replicated backend nodes. As shown in the below figure.



1.1. Scenario-1

Because of some reason the Pgpool-II node 3 encounters an issue with the connection to PG-STANDBY server. (P3-B2). As depicted in the next Figure.



Temporary Failure of Link or connection

Now when this temporary outage of P3-B2 link will be detected by and Pgpool-II node 3, it will start the failover of PG-STANDBY, And since the Pgpool-II is also part of the watchdog cluster so it will also inform the other two Pgpool-II nodes to start failover of PG-STANDBY PostgreSQL server and eventually the PG-STANDBY will be kicked out of the cluster.

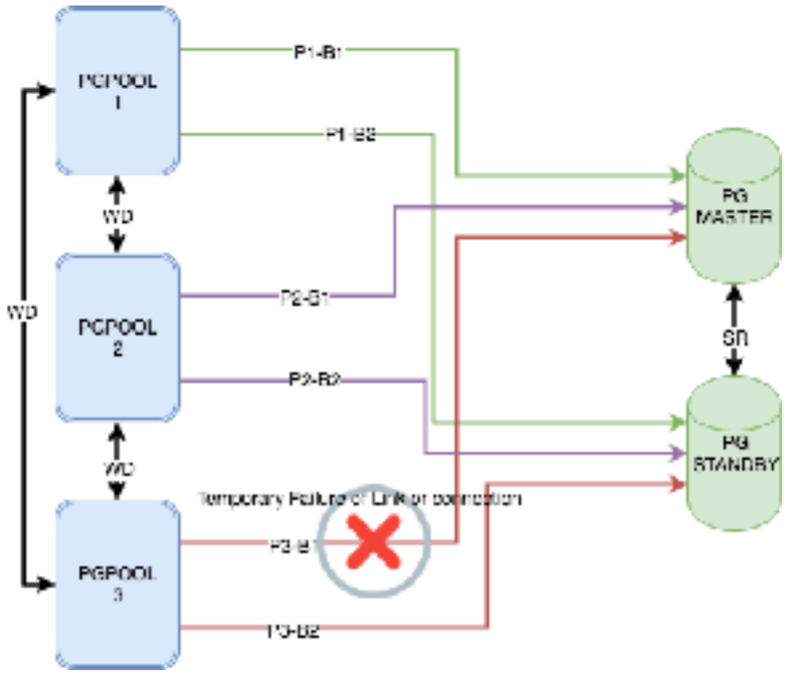
This failover of the PG-STANDBY can be termed as the false failover because the PG-STANBY node was actually in the healthy state, and it was also reachable by Pgpool-II node 1 and Pgpool-II node 2 when Pgpool-II node 3 started the failover on it because of P3-B2 link disruption.

1.1.1. What will happen in this situation with purposed quorum aware failover feature.

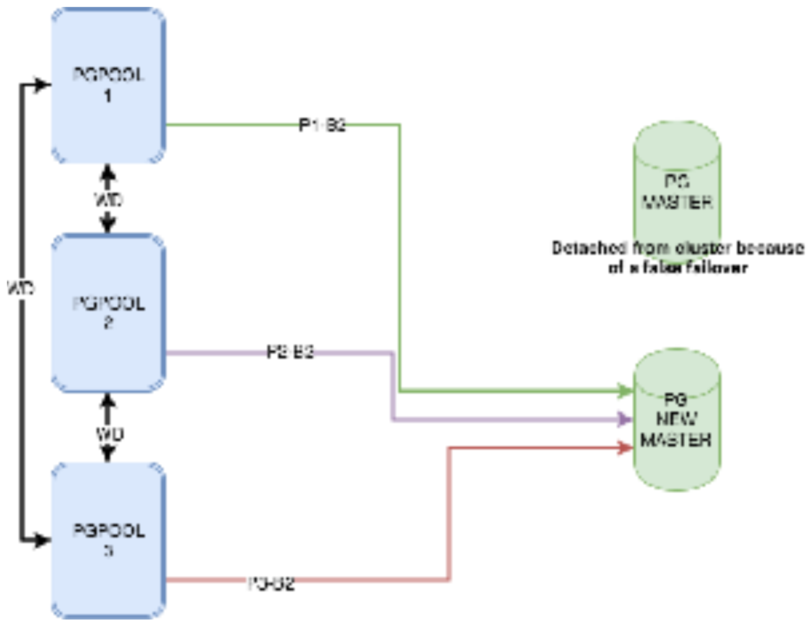
If we have the purposed quorum aware failover feature implemented in the Pgpool-II when the Pgpool-II node 3 detects the disruption in P3-B2 link (refer to figures above) it will ask the clutter to vote on the status of PG-STANDBY node and if the result of voting comes out in favour of PG-STANDBY that it is reachable by all other nodes then Pgpool-II node 3 will not perform the failover but change the status of the node to unreachable, and perform the local partial failover to make sure all client connections using standby PostgreSQL node should stop using it until it becomes reachable again.

1.2. Scenario-2

Consider again because of any reason the Pgpool-II node 3 encounters an issue with the connection to PG-MASTER server. (Link:P3-B1). As depicted in the next Figure.



Now when this temporary outage of P3-B1 link will be detected by and Pgpool-II node 3 it will start the failover of PG-MASTER, And as was in the scenario 1 (section 1.1) this failover will also be propagated to all attached Pgpool-II nodes (Pgpool-II node 1 and Pgpool-II node 2). And consequentially the PG-STANDBY node will be promoted to the new Master node and the original PG-MASTER will be detached from the clutter. As shown in the below figure.

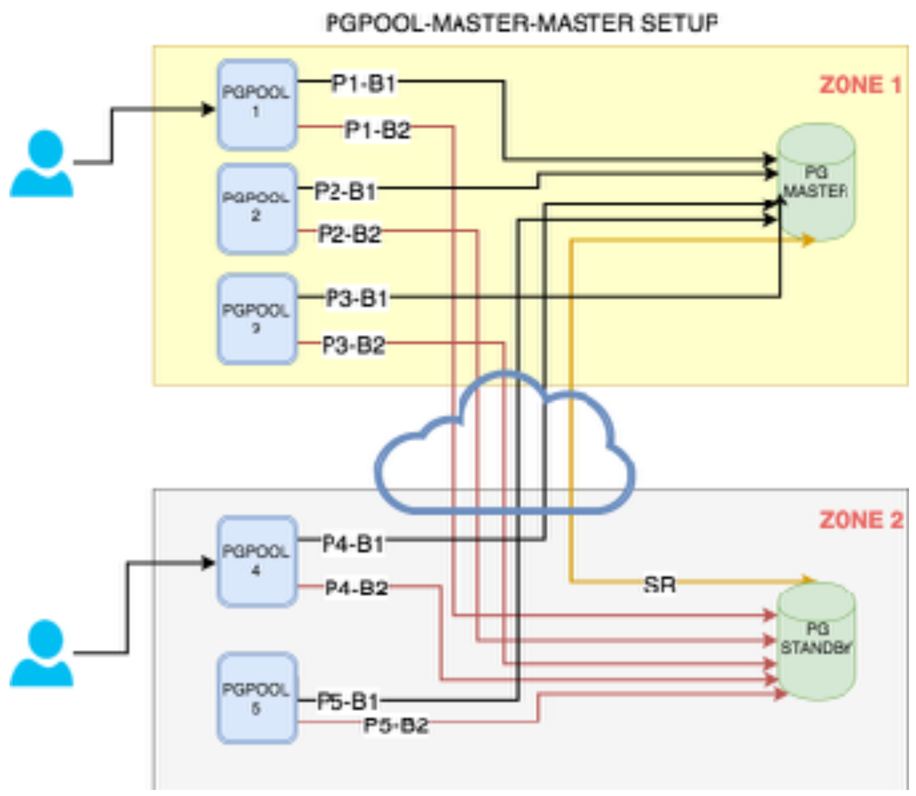


1.2.1. What will happen in this situation with purposed quorum aware failover feature.

Similarly as described in the section 1.1.1 If we have the purposed quorum aware failover feature implemented in the Pgpool-II when the Pgpool-II node 3 detects the disruption in P3-B1 link, it will ask the clutter to vote on the status of PG-MASTER node and if the result of voting comes out in favour of PG-MASTER then Pgpool-II node 3 will not perform the failover.
But unlike in section 1.1 where the Pgpool-II would just set the status of the backend PostgreSQL node to unreachable (Since it was a standby server and we could route all requests to master when the standby is not available) this time Pgpool-II node 3 will perform a local failover and will go into hibernate state (disallow all incoming connections and kill all existing user connections) until it's link with the master PostgreSQL is resolved.
Also in this case if the Pgpool-II node 3 was the master watchdog node it will resign from the watchdog master and let some other node to takeover the watchdog master node responsibilities.

2. Network Partitioning

Consider the Cluster design where Pgpool-II and PostgreSQL nodes are distributed in two network availability zones.

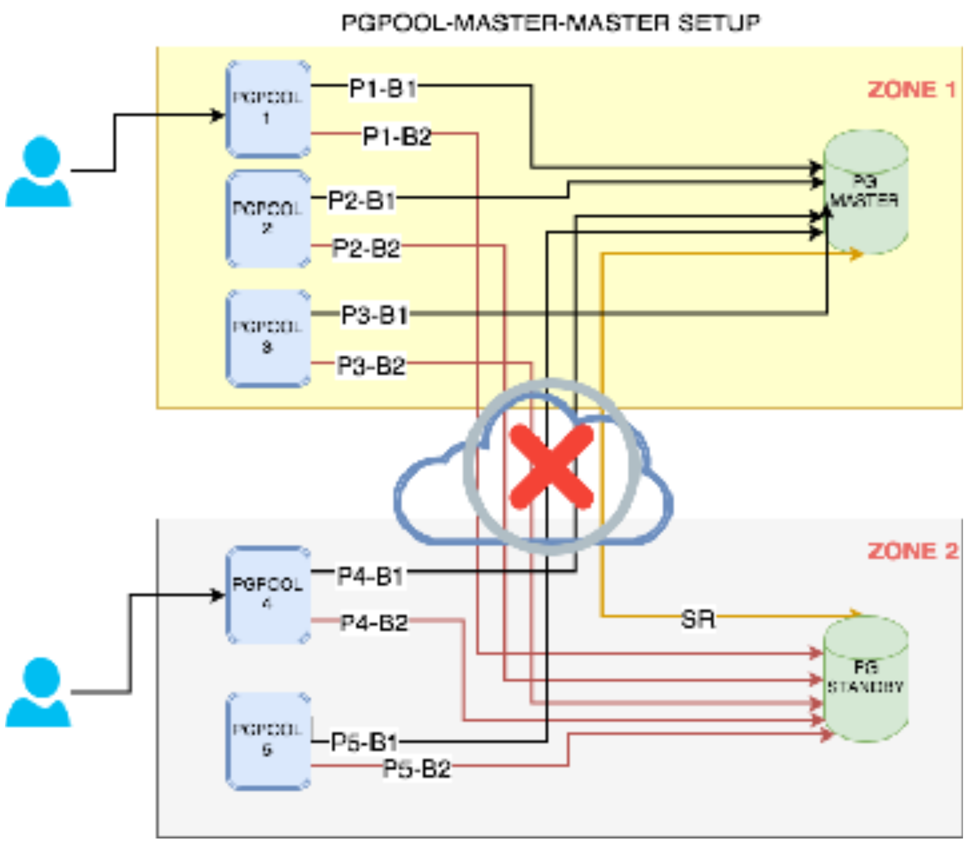


All Pgpool-II nodes (From Pgpool-II node 1 to node Pgpool-II node 5) are part of the same watchdog cluster but to save the clutter in the figure the watchdog connections are not shown.

As shown in the above picture this scenario consists of five Pgpool-II nodes and two streaming replicated PostgreSQL backend nodes.
The cluster is distributed in two network zones and both zones are connected through public network or a corporate link.
Note: Since the cluster consists of five Pgpool-II nodes so to complete the quorum the watchdog cluster would require consent of at least three Pgpool-II nodes.

2.1. Situation When ZONES become Isolated

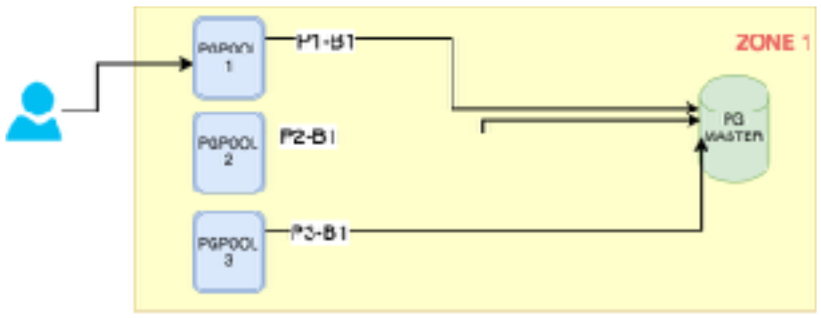
Now consider the situation where the ZONE-1 and ZONE-2 becomes isolated because of some public network/ corporate link issue.



After the problem in the link between ZONE 1 and ZONE 2 the network will be divided into two segments, and we will have two isolated watchdog clusters.

2.1.1. Cluster in Zone 1 (CLUSTER-1)

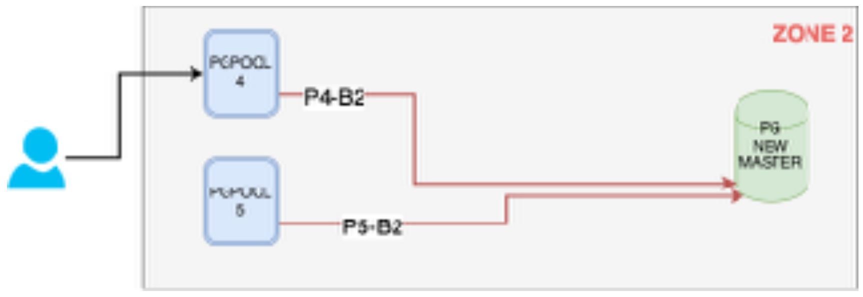
Once the network partitioning happens the Pgpool-II nodes in Zone-1 will perform failover of PostgreSQL node in zone-2 (PG-STANDBY) and detach it from the Pgpool-II nodes present in Zone-1.
The Zone-1 will look like as shown in the below picture and will this cluster will have following properties.



- The cluster will have three Pgpool-II nodes.
- The cluster will keep working with one PostgreSQL node and that PostgreSQL node will still keep its master PostgreSQL status.
- The watchdog cluster will still hold the quorum as it still has three out five reachable watchdog nodes.

2.1.2. Cluster in Zone 2 (CLUSTER-2)

Once the network partitioning happens the Pgpool-II nodes in Zone-2 will perform failover of PostgreSQL node in zone-1 (PG-MASTER) and detach it from the Pgpool-II nodes present in Zone-2.
The Zone-2 will look like as shown in the below picture and will this cluster will have following properties.



- The cluster will have two Pgpool-II nodes.
- The PostgreSQL node in Zone 2 will be promoted to the new master.
- The cluster will keep working with one PostgreSQL node which was promoted to master after the network partitioning.
- This watchdog cluster has only two alive Pgpool-II nodes out of total five, so it does not holds the quorum.

2.1.3. Problem.

After the cluster is divided into two sub clusters and each cluster has its own master PostgreSQL node, Whenever a write SQL is issued to the Pgpool-II in zone-2, we will end up with two different versions of data in both PostgreSQL servers, and that will require lots of manual work to recover it after rectifying the network partitioning.

2.2. What will happen in this situation with purposed quorum aware failover feature.

If we have the purposed quorum aware failover feature implemented in the Pgpool-II when the network partitioning happens the clusters will have the following properties.

2.2.1. Cluster in Zone 1 (CLUSTER-1) With purposed feature

The CLUSTER-1 in ZONE-1 will look and behave exactly same as it would have without the purposed feature (As in section 2.1.1). Reason being it still has the three reachable Pgpool-II nodes required to complete the quorum.

2.2.2. Cluster in Zone 2 (CLUSTER-2) With purposed feature

After the network partitioning with the purposed feature the cluster will behave as follows.

- Pgpool-II nodes in Cluster-2 (Pgpool-II node 4 and Pgpool-II node 5) will not promote the PG-STANBY (PostgreSQL server in ZONE-2) to the master PostgreSQL because they do not have the quorum to do so.
- Since the Pgpool-II nodes in the Cluster-2 does not have the reachable master PostgreSQL server so they will become hibernate and disallow all connections until the link is resolved.

2.2.3. Advantages

- Though the Pgpool-II service will become unavailable in Zone-2 but this way we will never end-up with multiple versions of data.
- Once the link between the Zones is restored, it is very easy to bring back the cluster to original state.