

Design for the integration of PostgreSQL logfile shipping into Sun Cluster.

The design goal of this project is to eliminate the need for shared storage in a cluster when using PostgreSQL Databases, and to allow WAL (log file) shipping between clusters.

If no shared storage is present in a cluster the underlying PostgreSQL database needs to be replicated from one node to the other. If a cluster has shared storage this is obviously unnecessary, but as soon as no shared storage is present the replication of the data is a must, to achieve high availability. An other scenario where it is desirable to work with logfile shipping instead of shared storage, is when campus clusters are stretched over long distances. In this cases the latency of write requests may be higher than acceptable, so log file shipping will be appropriate to solve a principle performance bottleneck.

On the other hand it will be possible to replicate the WAL file to a different postgres resource regardless wether it is in the same cluster or in a different cluster.

PostgreSQL offers the replication of its WAL files (redo log files) of the current node (primary node to an other node (standby node). The standby node is running in the recovery mode and applies the arriving WAL files to its database. To use this feature the `archive_command` needs to be configured at least on the primary node.

Whenever the primary completes an WAL file, he ships it to the standby and continues on the next WAL file. While the standby is applying the new arrived log to its database the primary continues on his current log. This concept implies, that the standby does not have the information of the actual log, but this is common with all other database like Oracle and Informix which offer similar concepts. To summarize this here, as soon as the standby is converted to a primary, the old primary is obviously out of sync.

Design Goals

The project will enhance the capabilities of the PostgreSQL agent in two scenarios:

1. Remove the requirement for shared storage between two nodes (shared storage replacement scenario).
2. Allow the Postgres agent to support a database which is running in recovery mode, and accepts log files from a different resource/cluster (recovery scenario). Typical use cases for this scenario are databases which have to be cloned from a consistent image and the clone can be used for backups, or intensive queries.

Storage replacement scenario

The integration of this WAL file shipping into the cluster has some prerequisites on the OS and the PostgreSQL site.

Prerequisites:

1. The nodes can communicate over ssh.
2. The PostgreSQL user trust each other in a way, that each of them can log in to the other

- node without a password.
3. The minimum PostgreSQL version is 8.2.x.
 4. The utility pg_standby is installed from PostgreSQL which is not part of a standard PostgreSQL build.
 5. The a remote copy over ssh in example the rsync utility is available.

This project has to achieve the goals listed below for storage replacement scenario.

1. On two nodes of the cluster each node will run a PostgreSQL database. One in the primary mode, and one as a standby.
2. If the primary node fails, the standby will be promoted as a primary.
3. The Sun Cluster PostgreSQL agent will prevent that the Sun Cluster framework will start both nodes as a primary by its automated processes.
4. The Sun Cluster PostgreSQL will prevent an amnesia situation. It should never be possible that as a result of an automated reaction the following scenario can occur.
 - Primary and standby are running on normal operation.
 - The primary fails and stays down.
 - The standby is promoted to a primary.
 - After some time the standby node goes down.
 - The primary comes up and acts as a primary but on its old and now invalid data.
5. Both nodes are acting as a primary in a split brain scenario.
6. If the standby fails and is restarted it will catch up on the WAL logs which were created in between.
7. Sun Cluster protects against the failure of the primary, as long as the standby is running.
8. Sun Cluster protects against the failure of the standby regardless if the primary is running or not.
9. Sun Cluster will deliver example scripts to resilver an outdated primary. These scripts can be customized to meet the SLA' in the customers environments.

The error scenarios listed in 3, 4 and 5 will result in two databases containing invalid information, and will be prevented. To achieve the goals above the following restrictions will apply.

1. After a failover to the standby happened, the failback is a manual operation, because the old primary is out of sync. This procedure needs to be manual, because the time to recreate the former primary is not deterministic.
2. The primary will not start if no standby is available. It is enough, that the standby is answering on ssh, and is configured to come up as a standby. It is not necessary, that the standby node runs the database at this time, but if it runs, it will be verified, that it acts as a standby. This restriction is necessary to prevent split brain and amnesia situations. If the administrator decides to start the primary, it is up to him to do this manually based on his informed decision.
3. Sun Cluster can not prevent the administrator to corrupt the database by manual interventions.

The PostgreSQL agent will be enhanced by a rolechanger component, which purpose is to trigger a conversion from a standby to a primary. During the conversion process The rolechanger will prevent the startup of the old primary. After the conversion the primary detects on its own that he is not allowed to start.

A typical resource group layout will be:

Two resource groups containing a PostgreSQL database resource only. One database resource acts as the designated primary and one as the designated standby. There will be another resource group

which will contain a logical host and a rolechanger resource. The rolechanger will have a strong dependency on the logical host resource and weak dependencies on both of the database resources. Because of this dependency requirement there will be no backport to Sun Cluster 3.1.

Configuration with zones controlled by the HA Container agent.

It is possible to configure the PostgreSQL agent to run on top of a zone controlled by the HA Container agent. In this case you would configure 2 resource groups, each of them controlling a zone in example zonea and zoneb, zone a would only be hosted in node1 and zoneb would only be hosted in node2. In the same resource group you will configure the PostgreSQL agent using the failover zone specific variable in the configuration file. Then you will have a resource group with the nodelist node1:zonea,node2:zoneb where you configure the rolechanger resource and the logical host. Due to the fact, that the rolechanger needs to float between the nodes and should not be tied to a specific zone on a node, it will not have failover zone specific configuration parameters.

Recovery scenario

The following will be achieved in the recovery scenario:

1. The PostgreSQL agent of the primary will tolerate the configuration of logfile shipping.
2. The PostgreSQL agent will tolerate to run in constant recovery mode on the standby. In this case the agent will perform process monitoring only.
3. If a standby is manually promoted to a primary, the PostgreSQL agent will silently switch to full monitoring, without manual intervention or interruption.

In the recovery scenario, there will be no automatic promotion from a standby database to a primary database.

There should not be a rolechanger resource configured in the recovery scenario.