GoldenGate vs. Data Guard

Edward Whalen
2019-04-10

I have seen a lot of questions concerning the difference between GoldenGate and Data Guard. Here is my shot about explaining the core differences.

Let's start with Data Guard. As blocks are changed in the database records are added to the redo log. Depending on the mode that you are running these log records will either be immediately copied to the standby or deferred. These are all changes to the database. Since they are log records and block data they are very quickly applied to the standby system. In the event of a failover to the standby recovery can occur very fast. Oracle has put decades of effort into optimizing the recovery process. All non-committed transactions are rolled-back, all committed transactions are rolled-forward. Recovery can happen in a matter of seconds.

GoldenGate works in a completely different mode. The GoldenGate extract process mines the redo log, keeping transactional changes in memory until a commit record has been processed. It then builds its own transactions into the trail file based on primary key. This can sometimes take time.

For example:

Source	Trail File
Update table set flag=1 where <criteria>;	Update table set flag=1 Where primary_key=<value>;
Updates 500,000 rows	Does 500,000 update statements based on primary key

Depending on the type of SQL statement this can consume some significant time.

GoldenGate is awesome for replication, especially lots of small transactions, but can be challenged by very large transactions. The recovery time for a "failover" can be significant compared to Data Guard. In addition, whereas you can configure Data Guard to work synchronously, GoldenGate only replicates changes after the transaction is committed, so if you have a long running transaction it can take a while to replicate.

I am a huge fan of GoldenGate and blog about it weekly, but if you are purely looking for synchronous DR then Data Guard is the best solution.

If you are looking to do any of the following the GoldenGate is the best solution:

- Replicate one or many tables to a read-write database.

- Replicate and transform tables.
- Bi-directional replication.
- Zero or near-zero downtime upgrade.
- Heterogeneous replication.