

Defining Multiple Replicat Processes to Increase GoldenGate Performance

Oracle states that GoldenGate can achieve near real-time data replication. However, out of the box, GoldenGate may not meet your performance requirements.

The GoldenGate documentation states "The most basic thing you can do to improve GoldenGate's performance is to divide a large number of tables among parallel processes and trails. For example, you can divide the load by schema".

But what if you have some large tables with a high data change rate within a source schema and you cannot logically separate them from the remaining tables due to referential constraints? GoldenGate does provide a solution to this problem by "splitting" the data and not the schema via the @RANGE function.

The Replicat process is typically the source of performance bottlenecks because, in its normal mode of operation, it is a single-threaded process that applies operations one at a time by using regular SQL. Therefore, to leverage parallel operation, the more Replicats the better (dependant on the number of CPUs on the target system).

The RANGE Function

The way the @RANGE function works is it computes a hash value of the columns specified in the input. If no columns are specified it uses the table's primary key. GoldenGate adjusts the total number of ranges to optimise the even distribution across the number of ranges specified. This concept can be compared to Hash Partitioning in Oracle as a means of dividing data.

With any division of data during replication, the integrity is paramount and will have an effect on performance. Therefore, tables having a relationship with other tables in the source schema must be included in the configuration. If all your source schema tables are inter-related, it is possible to include all tables!

The example below is from Oracle GoldenGate Reference Guide version 10.4:

(Replicat group 1 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (1, 3, ID));
```

(Replicat group 2 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (2, 3, ID));
```

(Replicat group 3 parameter file)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (3, 3, ID));
```

In my example in the next section, I include 3 tables in the source schema and walk through the complete configuration from start to finish.

Adding Replicats with @RANGE Function

I currently have an existing Replicat process on my target machine (linuxserver1) named RTARGET1 that includes the following 3 tables:

```
ORDERS
ORDER_ITEMS
PRODUCTS
```

My source database schema name is SRC and target schema TGT.

The following steps add a new Replicat named RTARGET2 with the relevant configuration and adjusts Replicat RTARGET1 parameters to suit.

N.B. Before conducting any changes, first stop the existing Replicat processes.

1. Check existing Replicat process is running.

```
GGSCI (linuxserver1) 61> info all
```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	RTARGET1	00:00:00	00:00:12

2. Stop existing Replicat process.

```
GGSCI (linuxserver1) 68> stop REPLICAT RTARGET1
```

```
Sending STOP request to REPLICAT RTARGET1 ...
Request processed.
```

3. Add the new Replicat process, using existing trail file.

```
GGSCI (linuxserver1) 69> add REPLICAT RTARGET2, exttrail ./dirdat/tb
REPLICAT added.
```

4. Now add the configuration by creating a new parameter file.

```
GGSCI (linuxserver1) 74> edit params RTARGET2
```

```
--
-- Example Replicator parameter file to apply changes
-- to target tables
--
--CHECKPARAMS
REPLICAT RTARGET2
SOURCEDEFS ./dirdef/mydefs.def
SETENV (ORACLE_SID= TARGET)
USERID ggs_admin, PASSWORD ggs_admin
DISCARDFILE ./dirrpt/rtarget2.dsc, PURGE
ALLOWDUPTARGETMAP
CHECKPOINTSECS 30
```

```

GROUPTRANSOPS 2000
-- set LOBWRITESIZE to 64K ckunks
DBOPTIONS LOBWRITESIZE 65536
-- This starts the macro
MACRO #exception_handler
BEGIN
, TARGET ggs_admin.exceptions
, COLMAP ( rep_name = "RTARGET2"
, table_name = @GETENV ("GGHEADER", "TABLENAME")
, errno = @GETENV ("LASTERR", "DBERRNUM")
, dberrmsg = @GETENV ("LASTERR", "DBERRMSG")
, optype = @GETENV ("LASTERR", "OPTYPE")
, errtype = @GETENV ("LASTERR", "ERRTYPE")
, logrba = @GETENV ("GGHEADER", "LOGRBA")
, logposition = @GETENV ("GGHEADER", "LOGPOSITION")
, committimestamp = @GETENV ("GGHEADER", "COMMITTIMESTAMP"))
, INSERTALLRECORDS
, EXCEPTIONSONLY;
END;
-- This ends the macro
REPERROR (DEFAULT, EXCEPTION)
REPERROR (DEFAULT2, ABEND)
REPERROR (-1, EXCEPTION)
REPERROR (-1403, EXCEPTION)
MAP SRC.ORDERS, TARGET TGT.ORDERS, FILTER (@RANGE (1,2));
MAP SRC.ORDERS #exception_handler()
MAP SRC.ORDER_ITEMS, TARGET TGT.ORDER_ITEMS, FILTER (@RANGE (1,2));
MAP SRC.ORDER_ITEMS #exception_handler()
MAP SRC.PRODUCTS, TARGET TGT.PRODUCTS, FILTER (@RANGE (1,2));
MAP SRC.PRODUCTS #exception_handler()

```

5. Now edit the configuration of the existing Replicat process. Add the @RANGE function to the FILTER clause of the MAP statement.

```

GGSCI (linuxserver1) 75> edit params RTARGET2
--
-- Example Replicator parameter file to apply changes
-- to target tables
--
--CHECKPARAMS
REPLICAT RTARGET1
SOURCEDEFS ./dirdef/mydefs.def
SETENV (ORACLE_SID=TARGET)
USERID ggs_admin, PASSWORD ggs_admin
DISCARDFILE ./dirrpt/rtarget1.dsc, PURGE
ALLOWDUPTARGETMAP
CHECKPOINTSECS 30
GROUPTRANSOPS 2000
-- set LOBWRITESIZE to 64K ckunks
DBOPTIONS LOBWRITESIZE 65536
-- This starts the macro
MACRO #exception_handler
BEGIN
, TARGET ggs_admin.exceptions
, COLMAP ( rep_name = "RTARGET1"
, table_name = @GETENV ("GGHEADER", "TABLENAME")
, errno = @GETENV ("LASTERR", "DBERRNUM")
, dberrmsg = @GETENV ("LASTERR", "DBERRMSG")
, optype = @GETENV ("LASTERR", "OPTYPE")
, errtype = @GETENV ("LASTERR", "ERRTYPE")

```

```

, logrba = @GETENV ("GGHEADER", "LOGRBA")
, logposition = @GETENV ("GGHEADER", "LOGPOSITION")
, committimestamp = @GETENV ("GGHEADER", "COMMITTIMESTAMP")
, INSERTALLRECORDS
, EXCEPTIONSONLY;
END;
-- This ends the macro
REPERROR (DEFAULT, EXCEPTION)
REPERROR (DEFAULT2, ABEND)
REPERROR (-1, EXCEPTION)
REPERROR (-1403, EXCEPTION)
MAP SRC.ORDERS, TARGET TGT.ORDERS, FILTER (@RANGE (2,2));
MAP SRC.ORDERS #exception_handler()
MAP SRC.ORDER_ITEMS, TARGET TGT.ORDER_ITEMS, FILTER (@RANGE (2,2));
MAP SRC.ORDER_ITEMS #exception_handler()
MAP SRC.PRODUCTS, TARGET TGT.PRODUCTS, FILTER (@RANGE (2,2));
MAP SRC.PRODUCTS #exception_handler()

```

6. Check both Replicat processes exist.

```
GGSCI (linuxserver1) 76> info all
```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	STOPPED	RTARGET1	00:00:00	00:10:35
REPLICAT	STOPPED	RTARGET2	00:00:00	00:12:25

7. Before starting both Replicat processes, obtain the log Sequence Number (SEQNO) and Relative Byte Address (RBA) from the original trail file.

```
GGSCI (linuxserver1) 78> info REPLICAT RTARGET1, detail
```

```

REPLICAT RTARGET1 Last Started 2010-04-01 15:35 Status STOPPED
Checkpoint Lag 00:00:00 (updated 00:12:43 ago)
Log Read Checkpoint File ./dirdat/tb000279 <- SEQNO
2010-04-08 12:27:00.001016 RBA 43750979 <- RBA

```

Extract Source	Begin	End
./dirdat/tb000279	2010-04-01 12:47	2010-04-08 12:27
./dirdat/tb000257	2010-04-01 04:30	2010-04-01 12:47
./dirdat/tb000255	2010-03-30 13:50	2010-04-01 04:30
./dirdat/tb000206	2010-03-30 13:50	First Record
./dirdat/tb000206	2010-03-30 04:30	2010-03-30 13:50
./dirdat/tb000184	2010-03-30 04:30	First Record
./dirdat/tb000184	2010-03-30 00:00	2010-03-30 04:30
./dirdat/tb000000	* Initialized *	2010-03-30 00:00
./dirdat/tb000000	* Initialized *	First Record

8. Adjust the new Replicat process RTARGET2 to adopt these values, so that the process knows where to start from on startup.

```
GGSCI (linuxserver1) 79> alter replicat RTARGET2, extseqno 279
REPLICAT altered.
```

```
GGSCI (linuxserver1) 80> alter replicat RTARGET2, extrba 43750979
REPLICAT altered.
```



Tip: Failure to complete this step will result in either duplicate data or ORA-00001 against the target schema, because GoldenGate will attempt to replicate the data from the beginning of the trail file (./dirdat/tb000000) if it exists, else the process will stall.

9. Start both Replicat processes. Note the use of wildcard (*)

```
GGSCI (linuxserver1) 81> start replicat RTARGET*
```

```
Sending START request to MANAGER ...  
REPLICAT RTARGET1 starting
```

```
Sending START request to MANAGER ...  
REPLICAT RTARGET2 starting
```

10. Check both Replicat processes are running.

```
GGSCI (linuxserver1) 82> info all
```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	RTARGET1	00:00:00	00:00:22
REPLICAT	RUNNING	RTARGET2	00:00:00	00:00:14

11. Check the detail of the new Replicat processes.

```
GGSCI (linuxserver1) 83> info REPLICAT RTARGET2, detail
```

```
REPLICAT RTARGET2 Last Started 2010-04-08 14:18 Status RUNNING  
Checkpoint Lag 00:00:00 (updated 00:00:06 ago)  
Log Read Checkpoint File ./dirdat/tb000279  
First Record RBA 43750979
```

Extract Source	Begin	End
./dirdat/tb000279	* Initialized *	First Record
./dirdat/tb000279	* Initialized *	First Record
./dirdat/tb000279	* Initialized *	2010-04-08 12:26
./dirdat/tb000279	* Initialized *	First Record

12. Allow users to connect to the source system and monitor the lag.

```
GGSCI (linuxserver1) 84> lag REPLICAT RTARGET*
```