ORACLE DATABASE 12c

Application Continuity

Checklist for preparation

ORACLE WHITE PAPER | MAY 2020

ORACLE

Contents

Application Continuity Checklist

The following checklist is useful for preparing your environment for using the Oracle Database 12c feature Application Continuity. Even if Application Continuity is not enabled on your database service, or not used by your applications, the points discussed here provide great value in preparing your systems' to support Continuous Availability.

Align Application and Server Timeouts

If an application level timeout is lower than timeouts provided for the detection and recovery times of the underlying system, then there is insufficient time available for the underlying recovery and replay to complete. Misaligned timers can result in replay by Application Continuity starting before the system has recovered, potentially causing multiple replays to be attempted before success, or worse still, requests completely timing out and an being error returned to your applications or users.

Consider that an application uses `READ_TIMEOUT` or `HTTP_REQUEST_TIMEOUT` or some custom timeout, then the following simple guidelines apply:

`READ_TIMEOUT` > `EXADATA` special node eviction (FDDN) (2 seconds in 12.1.0.2)

`READ_TIMEOUT` > `MISSCOUNT` (default 30 sec, modifiable in 12.1.0.2)

`READ_TIMEOUT` > Data Guard Observer: `FastStartFailoverThreshold` (default 30 sec, modifiable)

`FastStartFailoverThreshold` > `MISSCOUNT` (must be at least twice)

`READ_TIMEOUT` > `FAST_START_MTTR_TARGET`

`READ_TIMEOUT` > NET level: (`RETRY_COUNT`+1) * `RETRY_DELAY`

and

`READ_TIMEOUT` < `Replay_Initiation_Timeout` (modifiable on the service, default 300 seconds)

To avoid premature cancelling of requests the application timeout should be larger than the maximum of:

(`MISSCOUNT` (or FDNN) + `FAST_START_MTTR_TARGET`), (`FastStartFailoverThreshold` + `FAST_START_MTTR_TARGET` + TIME TO OPEN)

Configure the TNS or URL for High Availability

The following TNS/URL configuration is recommended for successfully connecting at failover, switchover, fallback and basic startup.
Set RETRY_COUNT, RETRY_DELAY, CONNECT_TIMEOUT and TRANSPORT_CONNECT_TIMEOUT parameters in the TNSnames or the URL to allow connection requests to wait for the service and connect successfully.

Set CONNECT_TIMEOUT to a high value to prevent login storms. Low values can result in 'feeding frenzies' logging in due to the application or pool cancelling and retrying connection attempts.

Do not set (RETRY_COUNT+1)*RETRY_DELAY or CONNECT_TIMEOUT larger than your response time SLA. The application should either connect or receive an error within the response time SLA.

These are general recommendations for configuring the connections for high availability. Do not use Easy Connect Naming on the client as EZCONNECT has no high availability capabilities.

This is the recommended TNS for ALL Oracle drivers for 12.2 and higher:

```
Alias (or URL) = (DESCRIPTION =
(CONNECT_TIMEOUT= 120)(RETRY_COUNT=20) RETRY_DELAY=3)
(TRANSPORT_CONNECT_TIMEOUT=3)
 (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
 (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

For JDBC connections in 12.1 the following should be used:

```
(DESCRIPTION =
(CONNECT_TIMEOUT= 15)   (RETRY_COUNT=20)(RETRY_DELAY=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
(ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

Enable Fast Application Notification (FAN)

FAN needs to be enabled. FAN is a mandatory component for interrupting the application to failover. When a node or network fails, the application needs to be interrupted in real time. Failing to enable FAN will lead to applications hanging when HARD physical failures occur. Hang times without FAN range from 8-15 minutes. Starting with Oracle Database 12c, there are three important enhancements for FAN:

- FAN is default, auto-configured and enabled out of the box with Oracle Real Application Clusters (RAC). FAN reads the URL and configures itself at the client. FAN is always configured at the Grid Infrastructure cluster. Using the URL format shown above is important for auto-configuration of FAN (using a different format will prevent FAN from being auto-configured).
- All Oracle clients use the Oracle Notification Service (ONS) as the transport for FAN
- FAN is posted by Global Data Services (GDS) to allow FAN events to span data centres.

Monitoring FAN

Use the `FANWatcher` utility to verify event posting and receiving: There is an updated (July 2017) version of `FANWatcher` available from OTN or via WebLogic Server Blogs. Refer to

Fast Application Notification -
http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/fastapplicationnotification12c-2538999.pdf

Monitoring FAN Events –
https://blogs.oracle.com/weblogicserver/monitoring-fan-events

UCP tracing can be enabled via the following procedure - to show FAN is being processed at UCP/Tomcat:

Logging can be configured using a properties file. The location of the properties file must be set as a Java property for the logging configuration file property. For example:

```
java -Djava.util.logging.config.file=myLog.properties
```

The logging properties file defines the handler to use for writing logs, the formatter to use for formatting logs, a default log level, as well as log levels for specific packages or classes. For example:

```
handlers = java.util.logging.ConsoleHandler
java.util.logging.ConsoleHandler.level = ALL
java.util.logging.ConsoleHandler.formatter =
java.util.logging.SimpleFormatter

oracle.ucp.level = FINEST
oracle.ucp.jdbc.PoolDataSource = WARNING
```

Feature specific logging can be enabled by setting a property in the logging configuration file. For all features:

```
clio.feature.all = on
```

or for specific features, such as:

```
clio.feature.conn_construction = on
clio.feature.high_availability = on
clio.feature.load_balancing = on
clio.feature.transaction_affinity = on
```

More information can be found in the Oracle documentation:

Universal Connection Pool Developer's Guide – Chapter 12 Overview of Logging in UCP

JDBC Developers Guide – Chapter 34 Diagnosability in JDBC

Steps for using Application Continuity

Return Connections to the Connection Pool

The application should return the connection to the connection pool on each request. It is best practice that an application checks-out a connection only for the time it needs it. Holding a connection when not in use is not good practice. An application should therefore check-out a connection and then check-in that connection immediately the work is complete. The connections are then available for subsequent use by other threads, or your thread when needed again. Following this practice also embeds request boundaries that Application Continuity uses to identify safe places to resume and end capture.

Use FAILOVER_RESTORE

Check if the application is presetting values on connections.
Some applications and mid-tiers configure connection pools such that all connections are, for example, in a preset language or time zone.

If session state is set intentionally on connections outside requests, and requests expect this state, replay needs to re-create this state before replaying.

Most common states are restored automatically by setting `FAILOVER_RESTORE` to `LEVEL1`.

Choose one of the following options:

- `FAILOVER_RESTORE=LEVEL1` set on the service
- Connection Initialization Callback for Java or the (older) TAF Callback for OCI
- Universal Connection Pool or WebLogic Server Connection Labeling

Starting with RDBMS 12.2, `FAILOVER_RESTORE=LEVEL1` is the recommended method.

Enable Mutable Use in the Application

Mutable functions are functions that can return a new value each time they are executed. Support for keeping the original results of mutable functions is provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `sequence.NEXTVAL`. If the original values are not kept and different values are returned to the application at replay, replay is rejected.

Configure mutable objects using `GRANT KEEP` for application users, and the `KEEP` clause for a sequence owner. When `KEEP` privilege is granted, replay applies the original function result at replay.

For example:

```
SQL> GRANT [KEEP DATE TIME | KEEP SYSGUID] … to USER
```

```
SQL> GRANT KEEP SEQUENCE mySequence to myUser on sequence.object
```

For JDBC Applications

Enable JDBC Statement Cache

For JDBC based applications, always use the JDBC statement cache for performance. When using Application Continuity, it is mandatory to disable a statement cache at the application server level (for example Tomcat, WebLogic, WebSphere etc) if used, as described in the JDBC reference documentation:

> If a statement cache at the application server level is enabled (for example, the WebLogic or third-party application server statement cache), this must be disabled when the replay is used. Instead, configure the JDBC statement cache, which performs better because it is optimized for JDBC and Oracle and because it supports Application Continuity.

Use `oracle.jdbc.implicitstatementcachesize=nnn`

Setting `nnn` to a positive value enables the Implicit Statement Cache.

Tune Garbage Collector

For many apps the default Garbage Collector tuning is correct. For an app with extreme performance requirements it may be necessary to perform JVM tuning. Our recommendation for extreme performance is to add the following to the Java command line (note that both attributes should be set to the same value):

```
java -Xms 2000m -Xmx 2000m
```

COMMIT

Application Continuity supports `COMMIT` of all styles: top-level, `AUTOCOMMIT` and `COMMIT` embedded in PLSQL. If your application is using a top-level commit, this is `OCOMMIT` or `COMMIT()` that is standalone, then there is full support for replay including when using `SESSION_STATE_CONSISTENCY=STATIC` mode (12.2.0.1). If your application is using `COMMIT` embedded in PLSQL or `AUTOCOMMIT`, it may not be possible to replay for cases where Application Continuity detects that the call including the `COMMIT` did not run to completion. Application Continuity will do the right thing.

For “top-level” `COMMIT`, for JDBC applications, you MUST disable `AUTOCOMMIT` either in the application or as property of `UCP`. This is particularly important when `UCP` is embedded in 3rd party application servers such as Apache Tomcat, IBM WebSphere and RedHat JBoss – if the application does not want `AUTOCOMMIT`.

Protection Levels

Please run the protection level trace report and concrete class analysis as per the Application Continuity white paper found at:

Application Continuity with Oracle Database 12c Release 2 :

Using AC Checking for Concrete Classes

and

Measure Coverage

(http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/overview/application-continuity-wp-12c-1966213.pdf)

Statistic monitoring beans are available in the ucpdemos.jar sample code at `http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html`:

Refer to the code samples `UniversalConnectionPoolManagerMBeanSample.java` and `UniversalConnectionPoolManagerSample.java` for information on MBeans.

Tracing server and JDBC replay feature trace will show coverage.

You must avoid using JDBC deprecated classes (concrete classes). This is a hard restriction in 12.1 but has been relaxed for some concrete classes in 18c. For information about the deprecation of concrete classes, including actions to take if an application uses them, see My Oracle Support Note 1364193.1 (`https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1364193.1`).

Diagnostics

Use the following in the “`OracleLog.properties`” file (should use absolute paths to the directory hosting the log files):

```
handlers = java.util.logging.FileHandler
java.util.logging.FileHandler.pattern = <REPLACE THIS PATH
FIRST>/jdbc%u.log
java.util.logging.FileHandler.limit = 10000000
java.util.logging.FileHandler.count = 90000
java.util.logging.FileHandler.formatter =
oracle.ucp.util.logging.UCPFormatter
.level = WARNING
oracle.jdbc.internal.replay.level = FINEST
oracle.ucp.jdbc.PoolDataSourceImpl.level = FINE
oracle.ucp.jdbc.oracle.level = FINEST
```

Tracing at the RDBMS server can be enabled via the `init.ora` file:

```
alter system set event='10602 trace name context forever, level
28:trace[progint_appcont_rdbms]:10702 trace name context forever, level 16:
41440 trace name context forever' scope = spfile ;
```

Additional Materials

OTN Home page for Application Continuity

http://www.oracle.com/technetwork/products/clustering/ac-overview-1967264.html

FAN whitepaper

http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/fastapplicationnotification12c-2538999.pdf

Application Continuity whitepaper

http://www.oracle.com/technetwork/database/options/clustering/application-continuity-wp-12c-1966213.pdf

Application Continuity OOW 2017

http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/hidingunplannedoutages-2872659.pdf

Client Failover

http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/client-failover-brief-2430007.pdf

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services

White Paper Title
May 2020
Author: Troy Anthony
Contributing Authors: Carol Colrain, Carmen Frank