Running Mission Critical Applications With IBM Db2®*

Andrew Hilden
Pandu Ranga Rao Mutyala
Francis Wong
Joyce Coleman
Kaya Gangesan
Hua Lu
{ahilden, pmutyala, fdewong, colemanj, gangesan, hualu}@ca.ibm.com

ABSTRACT

The fully managed database as a service has become a commodity for software developers and enterprises because it allows developers to focus on quickly creating applications while the cloud service provider takes care of the management and maintenance of the database. Reliability, availability, serviceability, and flexibility are some of the critical features required by developers and enterprises in a cloud service. IBM Db2®, a relational database management system, offers a cloud native, transactional managed service on IBM Cloud® that is designed to be fault tolerant to critical failures and scheduled maintenance, while also providing support for on-demand scaling of compute and storage resources [1] [2]. The IBM Db2® managed service on IBM Cloud® includes two key features that provide fault tolerance: a high availability (HA) option, and a disaster recovery (DR) option. When one or both of these options are used, data is kept in sync across multiple nodes or regions. Provided that developers modify their applications to leverage these features, users can access data seamlessly even during a service interruption. Without the need to write complex code or logic, developers can write mission critical applications that allow client connections to be rerouted using the Automatic Client Reroute (ACR) feature.

CCS CONCEPTS

• **Information systems → Data management systems**.

KEYWORDS

Managed cloud databases, Software applications, High availability, Disaster recovery, Fault tolerance, Cloud native

ACM Reference Format:
Andrew Hilden, Pandu Ranga Rao Mutyala, Francis Wong, Joyce Coleman, Kaya Gangesan, and Hua Lu. 2019. Running Mission Critical Applications With IBM Db2®. In *Proceedings of IBM CASCON (IBM CASCON'19).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

[0]IBM Db2® Managed Service on IBM Cloud®

IBM CASCON'19, NOV 2019, TORONTO, ON, CANADA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Enterprise database management systems must be highly available and resilient to disasters, critical component failures, and scheduled maintenance. Continuous availability of a database refers to the ability of applications connected to the database to handle connectivity loss or transaction exceptions when the database sustains an error. In this workshop, we describe how developers can enable continuous availability for their applications by leveraging the continuous availability features provided with the IBM Db2® managed database service on IBM Cloud®. The IBM Db2 HADR [3] feature allows users to configure a maximum of three standby nodes. In the Db2® managed database service, up to two standby nodes can be configured. First, the managed database service can be deployed with HA support, in which the standby is configured as a warm standby, and second DR support can be added. When a DR node is added, data is replicated geographically to an auxiliary standby in a remote region. The DR node can also be used as a read-only replica. When HA or DR is used, data is kept in sync through transaction log data captured on the primary that is shipped to the standby or standby(s). The standby(s) constantly roll forward the transaction logs to keep data in sync. Database availability can therefore be achieved with less downtime, making applications highly available and seamlessly available across nodes or regions.

2 DB2 DATABASE ENGINE INTERNALS

Each IBM Db2® HA service instance is deployed with two synchronous nodes and automation that continuously monitors the primary database. The automation initiates a failover to a warm standby node when a critical health check fails or a graceful takeover is initiated. The automation relies on a third-party consensus deployed outside of the service that determines which node is primary based on leader election. A publicly accessible floating IP deployed with the HA instance is moved to the new primary on failover. When a DR node is added to an existing HA cluster, data is asynchronously synced to the auxiliary standby node. When a DR node is added, the service enables a one-click failover app that allows an administrator to make the DR node primary if the HA pair becomes unavailable. The DR node is configured with a read-on standby IP that users can run read-only transactions against. A secondary DR IP is made available when the node becomes primary. Each HA and DR service instance is configured with an alternate server hostname and port that point to the floating IP provisioned on the HA cluster. When a data server client makes a connection, alternate server information is sent to the client by the server. On failover in-flight transactions

are rolled back and connections are rerouted to the new primary, allowing the client to receive specific SQL code exceptions. This allows the client to seamlessly move the connections over to the new primary as soon as it becomes available, and therefore continue to access data.

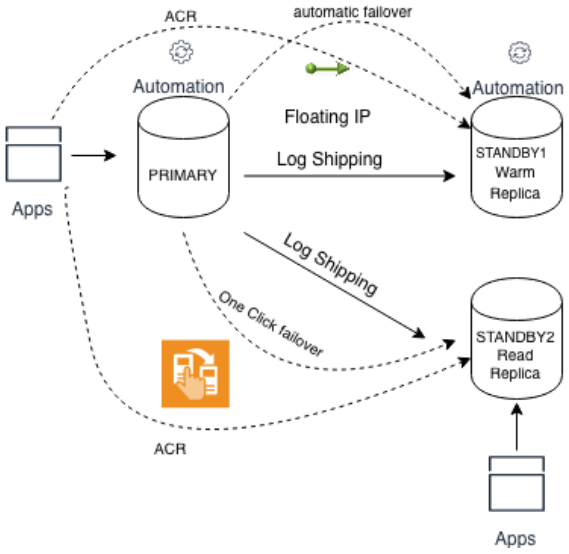

Figure 1: IBM Db2® HA and DR Setup

3 DB2 DATABASE DRIVER INTERNALS

Application availability is typically managed by allowing applications to hit multiple database server endpoints, and by rerouting connections to the primary database via a proxy. But with IBM Db2®, application availability is managed by rerouting the connections to the primary by using the alternate server hostname and port method. IBM Db2® drivers that support Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) have the ability to automatically reroute clients when a primary is relocated. The client reroute can be managed either through driver properties or with the configured alternate server hostname and port. After the first successful connection, the database server sends information to the client that contains the alternate server host and port. When a failover is initiated on the server side, the Automatic Client Reroute feature allows the client to recover from the failure by attempting to reconnect to the database using the alternate server and port. Required parameters include enableACR, enableSeamlessACR for Db2® CLI based connections, and for Db2® JCC Type 4 drivers clientRerouteAlternateServerName and clientRerouteAlternatePortNumber are required. For a non-Java based application, a SQL30108N error code is returned to the application, which indicates that a connection is re-established to the new primary, and any incomplete transaction is rolled back. If the application is not able to re-establish connection to the new primary, SQL30081N is returned, which the application should handle by attempting to reconnect to the new primary. Similarly for a Java-based application, a SQLJ4498N error code is returned when a connection is re-established, and on connection failure SQLJ4499N is returned. [4] [5].

REFERENCES

[1] IBM Cloud®. IBM Db2 on Cloud, 2018. https://cloud.ibm.com/catalog/services/db2.
[2] IBM Cloud®. HA in IBM Db2 on Cloud, 2018. https://cloud.ibm.com/docs/services/Db2onCloud?topic=Db2onCloud-ha.
[3] IBM Db2®. High availability disaster recovery, 2019. https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.admin.ha.doc/doc/c0011267.html.
[4] IBM Db2®. ACR Operation for Db2 Cli Apps, 2019. https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.apdv.cli.doc/doc/c0056198.html.
[5] IBM Db2®. ACR Operation for Db2 JCC Apps, 2019. https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.apdv.java.doc/src/tpc/imjcc_c0056197.html.

A DB2 DRIVER CONFIGURATIONS

A.1 Db2 CLI Driver Configuration for ACR

```
<configuration>
 </dsncollection>
 <databases>
  <database name="db" host="hostname" port="50001">
  <parameter name="keepAliveTimeout" value="20"/>
  <acr>
   <parameter name="enableAcr" value="true"/>
   <parameter name="enableSeamLessACR" value="true"/>
   <parameter name="maxAcrRetries" value="30"/>
   <parameter name="acrRetryInterval" value="1"/>
   <parameter name="enableAlternateServerListFirstConnect" value="true"/>
   <parameter name="affinityFailbackInterval" value="1"/>
   <alternateserverlist>
    <server name="S1" hostname="hostname" port="50001"/>
    <server name="S2" hostname="drhostname" port="50001"/>
   </alternateserverlist>
   <affinitylist>
     <list name="list1" serverorder="S1,S2"/>
   </affinitylist>
   <clientaffinitydefined>
     <client name="client1"
             hostname="localhost"
             listname="list1"/>
   </clientaffinitydefined>
  </acr>
  </database>
 </databases>
</configuration>
```

A.2 Db2 JCC Connect with ACR properties

```
java -cp db2jcc4.jar com.ibm.db2.jcc.DB2Jcc -url
"jdbc:db2://hostname:50001/BLUDB:sslConnection=true;
clientRerouteAlternatePortNumber=50001;
clientRerouteAlternateServerName=drhostname;
enableSeamlessFailover=1;
enableClientAffinitiesList=1;
maxRetriesForClientReroute=3;
retryIntervalForClientReroute=2;
allowNextOnExhaustedResultSet=1;"
-user <REDACTED> -password <REDACTED>
```