Keeping Databases in Sync during migration from z/OS to a distributed platform

Avijit Goswami, PMP®, ITIL®, IBM Certified DB2 DBA,
Sr. Technology Architect, Infosys Limited
avijit_goswami@infosys.com

Abstract

This paper is intended to show the different approaches that can be adopted to keep databases in sync for large scale Database migrations. The solutions documented in this article are based on the real life implementation of an Application Modernization and Re-hosting project that included migration of legacy DB2 z/OS databases to UDB on LUW platform for a large organization in the USA. Synchronizing the centralized databases during migration is one of the critical success factors for a large scale database migration project. This article shows how a hybrid approach, depending on the complexity of the environment, can be adopted to overcome synchronization challenges during different phases of migration. The solutions discussed in the article should help database architects to make an appropriate decision on how the migration groups can be formed and how solutions can be implemented to minimize the impact.

Introduction

Many IT organizations today are gradually moving away from their legacy systems, replacing legacy data stores (including DB2, IDMS, IMS, and VSAM etc.) with different RDBMS databases residing on distributed platforms and legacy applications with web based applications. Today, projects requiring database migration from DB2 for z/OS, IMS on z/OS or IDMS on z/OS to other RDBMS databases on distributed platforms are one of the top priorities for large organizations. These database migration projects help organizations simplify their technology environments by reducing the number of database platforms, taking advantage of the newest features on RDBMS platforms and finally reducing of cost of supporting multiple databases. However, when the legacy database is large and preserved as a centralized repository of mission critical data for diversified applications, it becomes immensely important that the integrity of data during different phases of application and data migration is guaranteed. For large scale application migrations, we generally create different logical groupings to reduce the complexity of the migration tasks and realize project objectives and goals in relatively shorter durations. This makes the task of synchronization of database objects more important during the whole life cycle of the project. Ignoring this aspect can contribute to project delay and in some cases failure of a data migration project. As mentioned in a Bloor Research white paper:

"Approximately 60 percent of data migration projects have overruns on time and / or budget, which affect business continuity and disrupt operations. . . Some projects fail completely."

So, identifying issues with data synchronization issues ahead of time and putting a proper plan together can make the data migration project a smooth sailing.

Problem Statement

For many organizations it is very common that there is a centralized legacy database which can be accessed by multiple applications to read and update the common data. It makes life more complex in terms of how we arrive with logical application migration groups where impacted application programs and database tables are grouped in such a way that they are totally isolated from any other application programs. In most of the cases in complex environments, we end up creating logical groups where one database (here DB2) table is accessed by application programs outside the group and vice versa: the application programs in the group access DB2 tables that are residing outside the migration group. This article provides a detailed approach on how we can overcome the data synchronization issues and meet the project objectives for large scale migrations.

Solutions

There can be multiple solutions that can be implemented to keep the databases in sync during the database migration. Some of the options are outlined below:

Option 1: Migrate Database and Application in a group with remaining Legacy Applications modified to use UDB

The following approach should be taken to migrate the impacted application code and databases to the new platform while changing the remaining legacy code to point to the migrated database instead of using DB2 on z/OS.

- Migrate the legacy application code and database to the new platform
    a. The migrated code will start using the database migrated to UDB on LUW
    b. If the migrated application needs to access any table which is not yet migrated, then DB2 Connect using DRDA access would be used to access the database still residing on z/OS
- Remaining applications that are yet to move to new platform, will access the migrated database using DRDA access
    a. A thorough impact analysis would be done to identify the impacted code
    b. The following coding method to be adopted for the impacted legacy code
        i. Three-part table names
        ii. Explicit connect statements
    c. The following precompiler options[1] to be used
        i. Use CONNECT(2), explicitly or by default
        ii. Use SQL(ALL) explicitly for a package that runs on a server that is not DB2 UDB for z/OS
    d. The following remote-bind DRDA access process should be considered
        i. Bind the DBRM into a package at the remote location.
        ii. Bind the remote package and the DBRM into a plan at the local site, using the bind option DBPROTOCOL(DRDA).

Advantages:

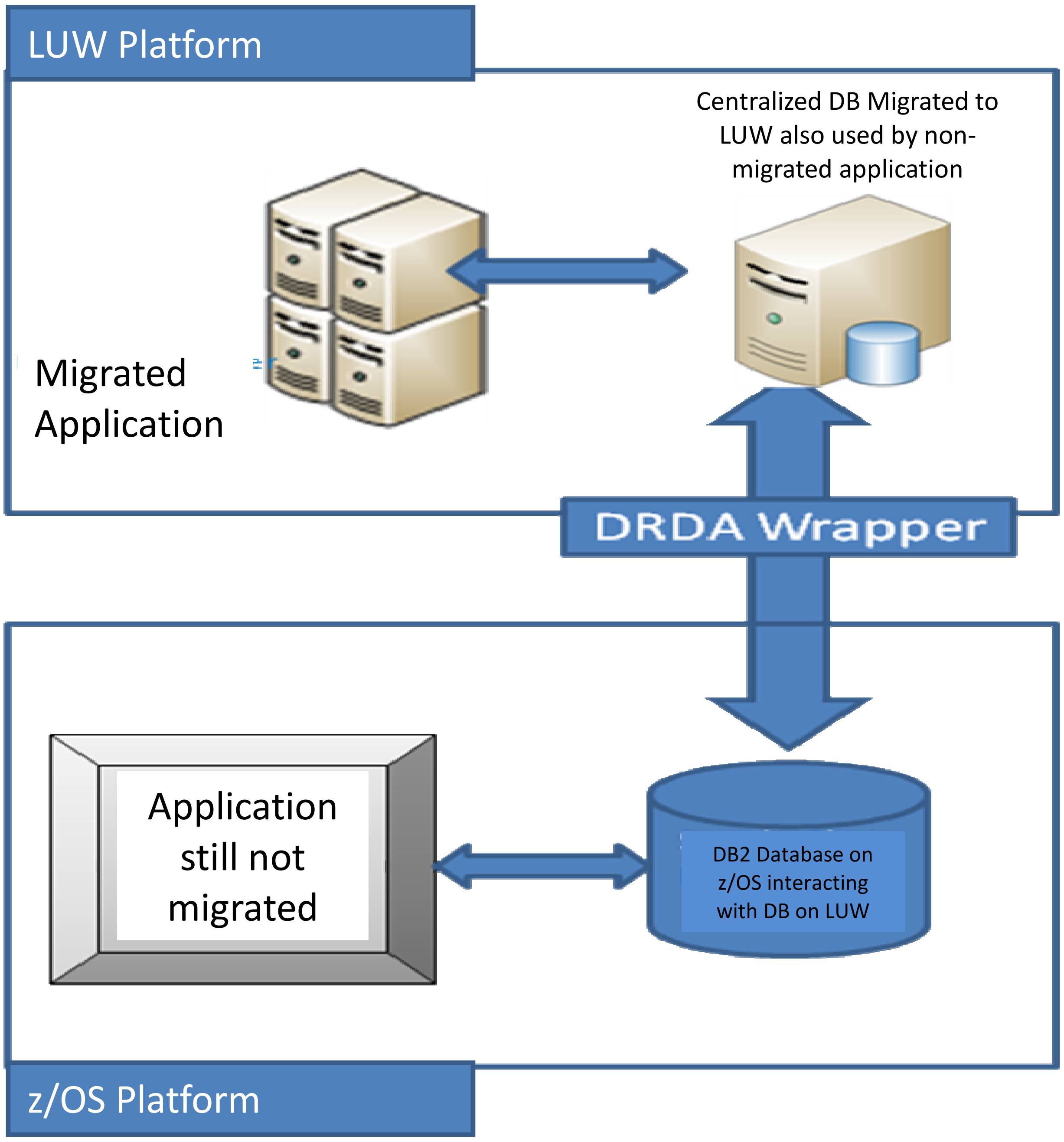
1. Single copy of the Database objects
2. No additional processing required to keep the Databases in sync
3. Minimal changes in the application code. Special attention on the Precompiler and Bind options to enable DRDA access

[1] For a comprehensive procedure for pre-compiler and bind options using DRDA, refer to http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.apsg/q41cpp.htm

Disadvantages:

1. Temporary changes in the Legacy code to Connect to the Remote servers
2. Slight increase in response time due to network overhead

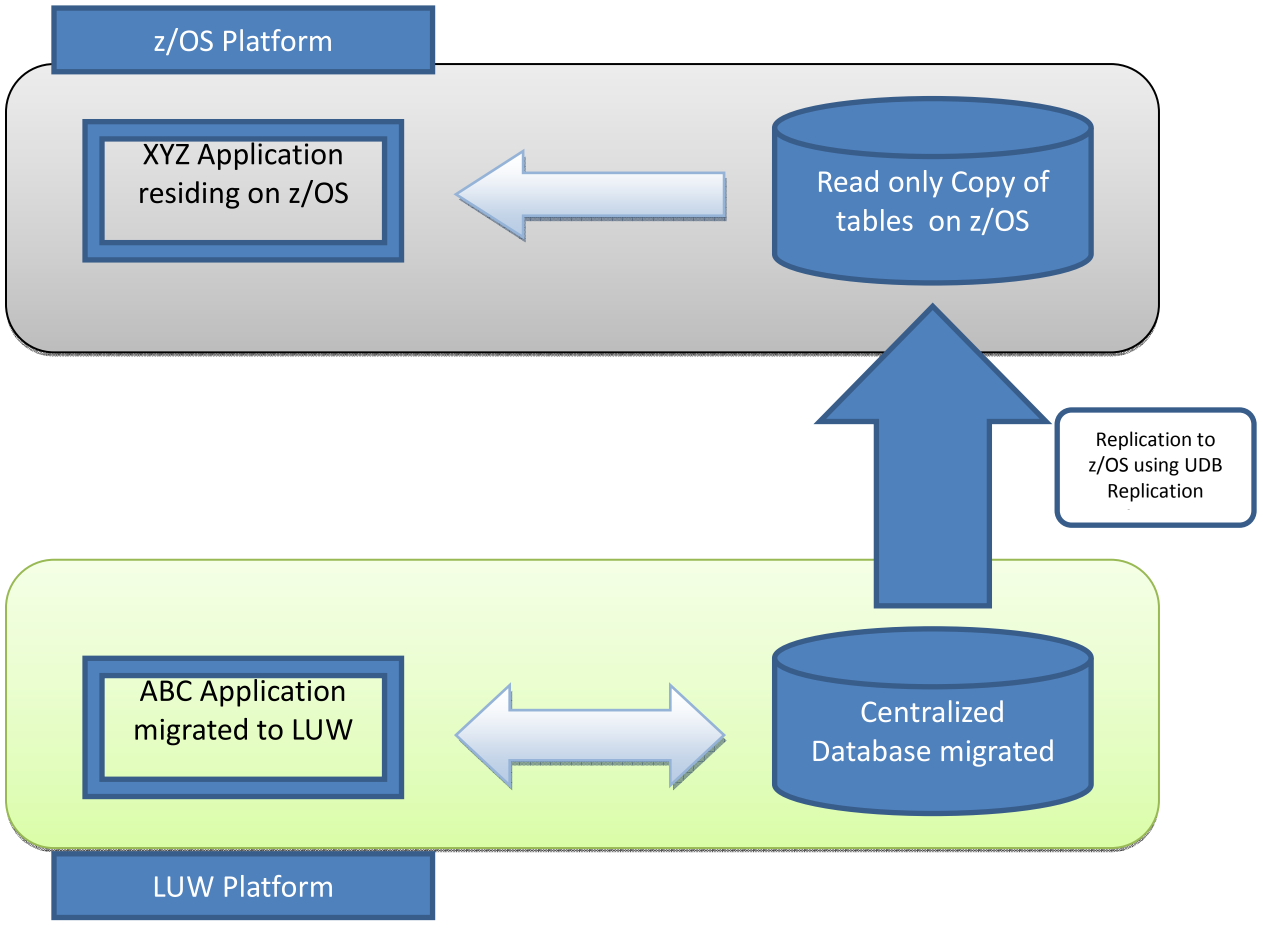
The DRDA Connection between DB2 UDB on LUW and DB2 on z/OS is shown in the diagram below:


LUW Platform
Centralized DB Migrated to LUW also used by non-migrated application
Migrated Application
DRDA Wrapper
Application still not migrated
DB2 Database on z/OS interacting with DB on LUW
z/OS Platform

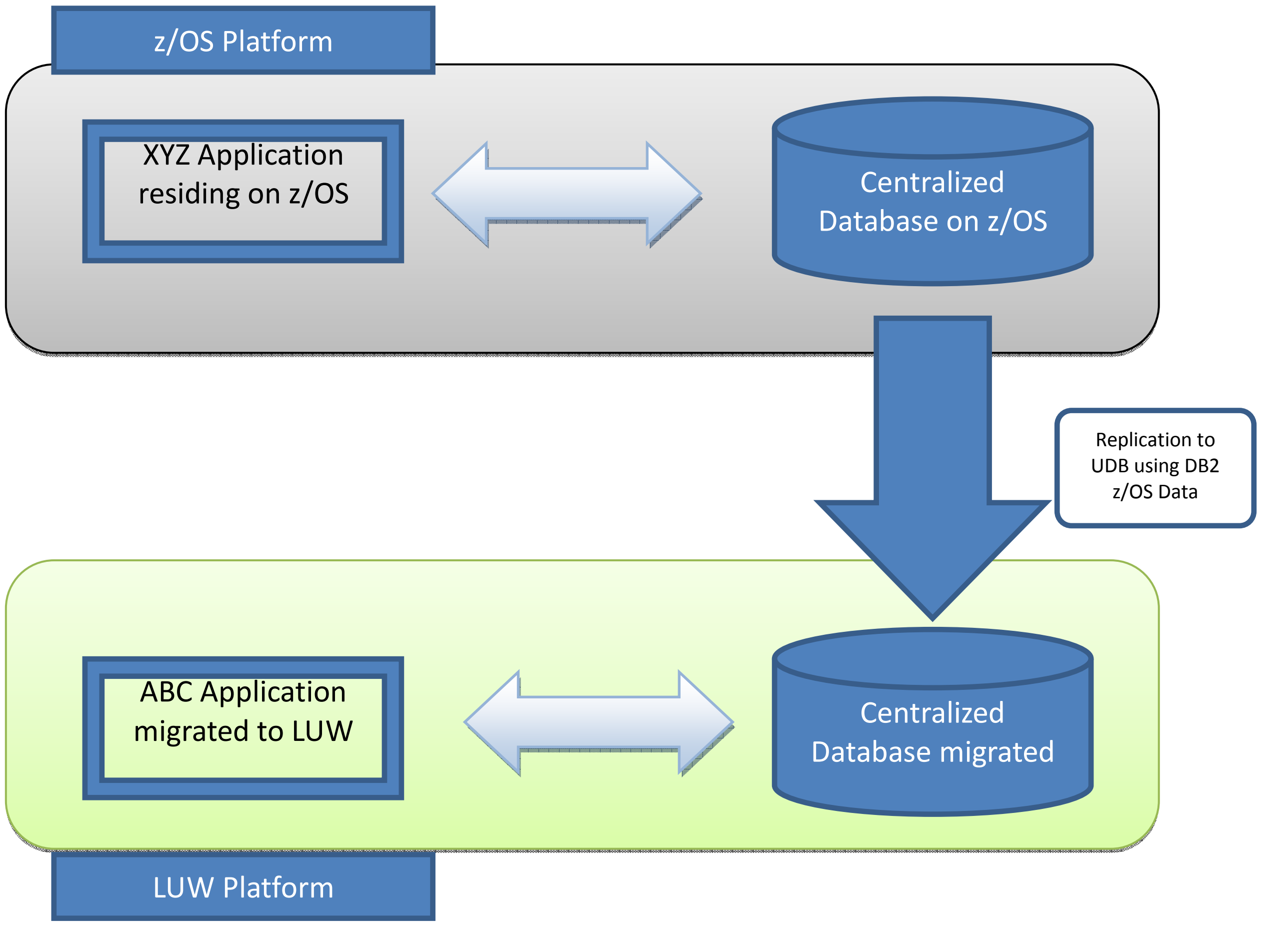
Option 2: Using DB2 Replication Centre on LUW and IBM DB2 Data Propagator/WebSphere Information Integrator for z/OS, Version 8.1 or later

In this tool based approach, the source and target tables would be synchronized using DB2 Replication capabilities. The following scenarios can be covered using this approach

A. The XYZ Application residing on z/OS is accessing a DB2 database for read only purposes. Two copies of the DB2 impacted objects should be retained; one in Mainframe and one in LUW. Any update to LUW UDB by ABC application should be replicated to z/OS copy of DB2 using UDB Replication Center See the diagram below:


z/OS Platform
XYZ Application residing on z/OS
Read only Copy of tables on z/OS
Replication to z/OS using UDB Replication
ABC Application migrated to LUW
Centralized Database migrated
LUW Platform

B. XYZ Application residing on z/OS accessing copy of DB2 objects to Update/Insert/Delete records. Updates to z/OS DB2 objects should be replicated to UDB using DB2 DataPropagator/WebSphere Information Integratorfor z/OS. This is shown in the diagram below.


z/OS Platform
XYZ Application residing on z/OS
Centralized Database on z/OS
Replication to UDB using DB2 z/OS Data
ABC Application migrated to LUW
Centralized Database migrated
LUW Platform

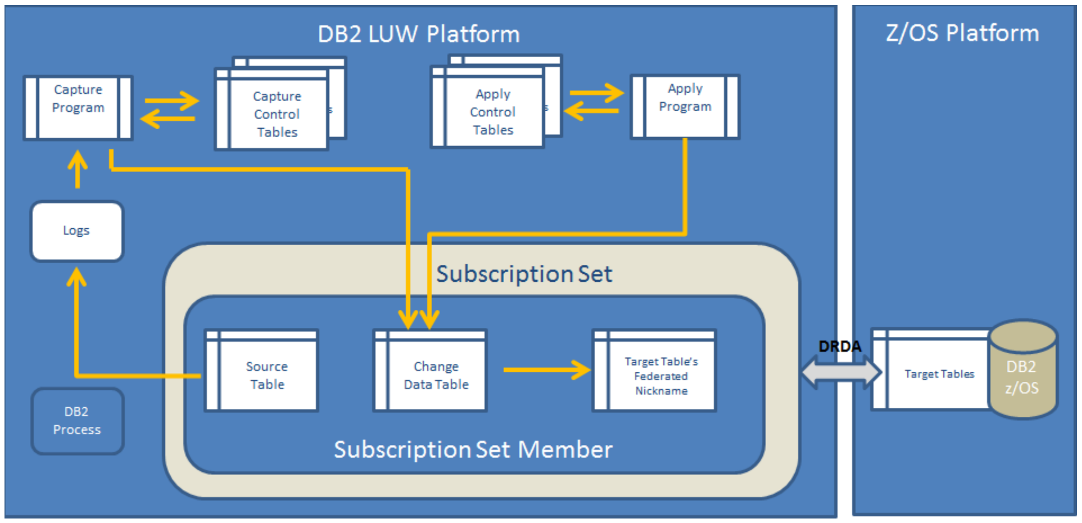
C. ABC application accessing any XYZ Database on z/OS platform can access the tables using Nicknames created in UDB federated server.

The scenario which combines A and B should be avoided by creating migration groups strategically to avoid two way replications. The groups should be created in such a way so that XYZ Application programs making updates to DB2 tables owned by the ABC Application should be migrated along with ABC Applications programs. Option B should only be used when it is absolutely unavoidable to migrate the XYZ code together with the ABC application code.

To create SQL replication from UDB LUW platform to DB2 on z/OS platform, the following steps are in general followed on the LUW platform

i. Create the Replication process related tables (Changed Data, Capture and Apply Control Tables) in LUW
ii. Create a federation server in LUW to connect to DB2 on z/OS
iii. Identify the z/OS tables where the changes will be replicated
iv. Create federated nicknames in LUW platform for the target z/OS tables
v. Create subscriptions sets in LUW and create mapping between source (LUW table) and target tables (Nickname in LUW)
vi. Activate the subscription sets.

The diagram below depicts this scenario.



Advantages:

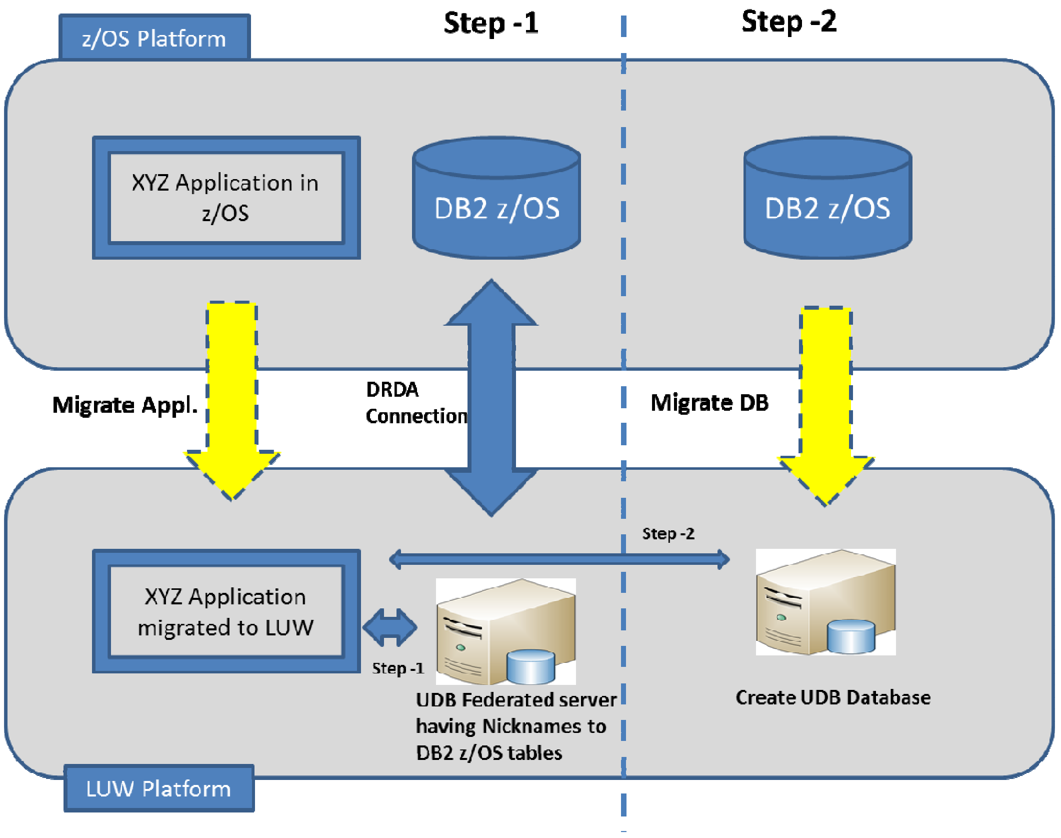
1. Minimum dependency on which server the database is residing when migrating the application
2. The replication utilities will minimize effort for manual designing and coding bridge programs to keep the databases in sync
3. Easier to activate and deactivate replication process on member/table level based on changing requirements

Disadvantages:

1. Two copies of tables would consume additional storage space
2. Need additional memory for replication if more data being replicated and the number of concurrent transactions is high
3. Additional log volumes are required for source table and change data tables
4. The transaction response time may increase for those transactions that heavily update source tables

Option 3: Database on Mainframe and Application Codes Migrated to new Platform

This approach is based on a phase-wise migration where in the first phase all applications would be migrated to the new server on distributed platform. The migrated application will access the database residing on Mainframe DB2 connectors (i.e. DB2 Type-4 connector). In the second phase of the project, all the databases will be migrated to LUW UDB. Connections to the databases will be changed for the migrated applications. See the diagram below.



Advantages:

1. Application and database migrations can be done independently
2. Complexity of application migration in terms of database dependencies can be reduced considerably
3. Could be a good approach when budget is a constraint and databases can be migrated outside application as a separate project

Disadvantages:

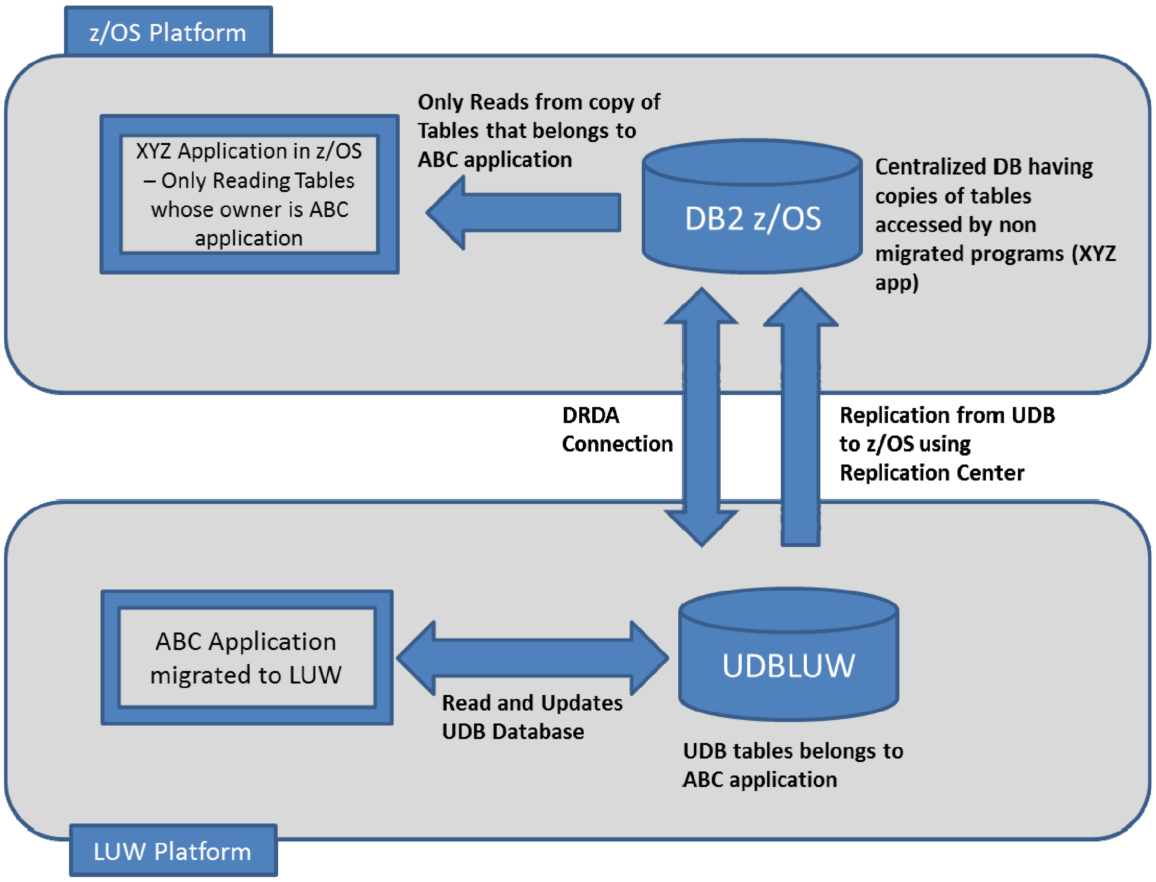
1. Network latency between Application and Database servers can become performance bottlenecks
2. Varied skillsets are required for the support engineers as application and database are in separate platforms
3. Application programs may need to be changed to use Host Variable Arrays and Multi Row fetch and inserts to reduce network latency

Option 4: Hybrid Approach

This approach deals with the most practical scenario of migrating the applications as logical groups to the new distributed platform along with the databases it owns (updates) in a phased manner. To ensure that the centralized databases are in sync and all applications getting the most updated data from DB2, following steps should be taken.

A. In the first step a logical grouping of the application code along with the tables it updates will be produced. Then this group (e.g. ***Group A***) of application code and database tables will be migrated to the distributed platform. Any other tables residing on z/OS which does not belong to this group will be accessed from distributed platform using DRDA Bind protocols.

B. In this step, the tables which are under ***Group A*** but still need to stay on z/OS platform as other application groups are reading data from these tables needs to be synchronized with UDB on LUW. This could be achieved using the UDB Replication Center.


z/OS Platform
XYZ Application in z/OS – Only Reading Tables whose owner is ABC application
Only Reads from copy of Tables that belongs to ABC application
DB2 z/OS
Centralized DB having copies of tables accessed by non migrated programs (XYZ app)
DRDA Connection
Replication from UDB to z/OS using Replication Center
ABC Application migrated to LUW
Read and Updates UDB Database
UDBLUW
UDB tables belongs to ABC application
LUW Platform

Advantages:

1. Reduced complexity in terms of keeping databases synchronized as the replication is needed only from LUW to z/OS side
2. Flexibility in grouping the application programs and makes the migration groups smaller. This will also reduce the project risk

Disadvantages:

1. Two copies of common tables
2. Additional storage
3. Network overhead for replication

Conclusion

Synchronizing the centralized databases during migration is one of the critical success factors for a database migration project. As documented in this article adoption of a hybrid approach based on the complexity of the environment is the best approach in most of the situations. The solutions discussed in this article should help database architects to make an appropriate decision on how the migration groups can be formed to minimize the impact.

References

- http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.apsg/q41cpp.htm
- http://supportline.microfocus.com/documentation/books/hc30/mmintr.htm
- http://cibecs.com/blog/2011/04/19/data-migration-best-practice/