Selected features of DB2 & metadatabase replication

Igor Yakushin
4/09/02

LIGO-T020058-00-L

Content

- db2cc
- Administration server
- Transaction logging
- Returning a database to a consistent state after minor failures
- Backup and recovery
- Backup and recovery: configuration at sites
- Cataloging remote databases for a client-server communication
- DB2 Discovery
- DB2 Authentication

Content

- Federated system
- Replication in DB2
- How to setup LDAS metadatabase replication

db2cc

- db2cc is Java GUI for DB2 administrator and user.
- In 99% cases various DB2 administration guides explain how to do things using db2cc (DB2 command center) without necessarily providing enough information about how to do it from a command line (that's especially true about replication).
- Using db2cc is quite convenient and saves a lot of time.
- db2cc can generate command line or SQL equivalent for almost everything it is doing. So if one needs a script to automate things, one can do it in the first approximation using db2cc and edit the generated script by hand later.

db2cc

- Creating/dropping/updating databases, tables, views, indexes, triggers, schemas.
- Loading, importing, exporting data.
- Backup/restore.
- Reorganizing data, collecting table statistics.
- Setting up replication between systems.
- Managing database connections (needed for remote database access, replication, federated databases).
- Monitoring resources and events (useful for database optimization).

db2cc

- Creating, executing, scheduling scripts (command center, script center, journal).
- Viewing all the details of the access plan chosen by the DB2 Optimizer for a given SQL statement (visual explain).
- Identifying how to fine-tune SQL statements (visual explain).

DB2 administration server (DAS)

- Most of the DB2 tools (including db2cc) require their own instance that operates concurrently with all the other (data) instances. Such special instance is DAS.
- It is also responsible for:
  - Communications between DB2 servers (necessary for federated database and replication configurations, for example);
  - Remote administration of DB2 servers
  - Managing, scheduling, executing user-defined DB2 and OS command scripts.

Transaction logging

- Transaction logging is used to restore data consistency in case of various hardware/software failures.
- Change to a row in a table -> log buffer -> log file.
- Insert row -> new row, update row -> both old and new row, delete row -> old row. Also commit, rollback are recorded.
- Records are moved from log buffer (memory) to log file after commit, rollback or when all log buffers are full.
- Certain type of transaction logging is needed to support online backups and replication.

Transaction logging: circular and archival

- Circular logging:
  - Specified number of primary logs of specified size are allocated on disk in advance whether needed or not when the database is created or reconfigured;
  - Primary logs are filled in a circular fashion;
  - Each log is marked 'reusable' once all records stored there are applied to the corresponding database by commit or rollback;
  - If a database runs out of reusable primary logs, secondary logs are used (not allocated in advance but created when needed up to the number specified in database configuration);
  - Secondary logs are released when there are enough reusable primary logs available;
  - Offline backups, no online backups, no replication.

Transaction logging: circular and archival

- Archival (retention) logging:
  - Primary logs are not reused but archived filling the disk until deleted;
  - Secondary logs are not used;
  - Three types of primary logs:
    - Active - have transactions that have not been committed or rolled back,
    - Online - completed transactions but stored together with active logs,
    - Offline - completed transactions, stored somewhere else,
  - Supports offline and online backups, rollforward, replication;
  - DBA must take care of storing logs needed for database restoration and removing the ones which are no longer needed.

Transaction logging: returning a database to a consistent state after minor failures

- Whenever some transactions are unexpectedly interrupted (by power outage or an application failure), the corresponding databases are placed in an inconsistent state.
- Such a database must be restarted (either automatically by database manager or manually by DBA depending on configuration). The restart procedure examines the log files and moves the database to the nearest consistent state.
- To make restart procedure faster, one might periodically insert soft checkpoints - points in time when the database was ordered to complete all the existing transactions before starting the new ones. The more checkpoints were made, the faster the recovery will be.

Backup and recovery

- Restart cannot handle more serious failures: corrupted storage media. Backup is needed.
- Backup types:
  - Offline
    - No other applications can access the database during backup,
    - The whole database must be backed up,
    - It can only be restored to the state it was when the backup was taken,
    - Does not need archival logs;
  - Online
    - Can be done in parallel with other applications,
    - Either the whole database or just some tablespaces can be backed up,
    - Later logs can be applied to an online backup image to restore the database to the end of logs or any other point in time,
    - Log retention (archiving) should be enabled.

Backup and recovery

- Restoring the content of the whole database can only be done offline, restoring tablespaces can be done online.
- If the original tablespaces are no longer available, one can do a redirected restore to different tablespaces.
- One can restore on a different machine but on the same platform (to test how it works and also to test whether I am keeping the correct logs, I restored LLO_1,2 backups at LHO and LHO_1,2 backups at LLO).
- If archival logging was enabled, one can use roll-forward recovery: first, restore the database from a backup image, second, apply logs to roll-forward to a specified point in time or to the end of logs.

Backup and recovery: configuration at sites

- Archival logging is enabled with 20 16Mb primary log files.
- Online backups are scheduled with cron (instead of internal DB2 scheduler that seems to have no command line interface) to run at 3:00am, 3:10am, 3:20am, 3:40am every day for 4 databases at each site. Each online backup takes 1-2 minutes for the biggest database we currently have (~ 2 Gb).
- Scripts for each backup are stored at ~ldasdb/bin/backup_${DB}.sh. Scripts used to restore LHO_2/LLO_2 databases at LLO/LHO are called ~ldasdb/bin/restore.${DB}.sh.

Backup and recovery: configuration at sites

- Since archival logging is used and backups are taken every morning, one must take care not to fill the disk space with old backups and logs that are no longer needed.
- Perl script ~ldasdb/bin/cleanBackupsLogs.pl is used to keep the last N=3 backups for each database, figure out which logs are no longer needed to roll-forward from each kept backup, and delete all the other backup images and log files. It is scheduled to run at 4am every day. It records its action to ~ldas/backup/cleanBackupsLogs.history log file. One can delete it when it becomes too large. 'db2 list history backup since $timestamp for $database' command is used to figure out which backup needs which logs for roll-forward recovery.
- TODO: take into account S9999999.log -> S0000000.log.

Cataloging remote databases for a client-server communication

- Before DB2 Database Manager can access a database, it must be cataloged (listed) in the system database directory file (same applies to systems and instances):
  - db2 list database directory
- When dealing with local databases, one rarely has to use catalog/uncatalog commands because they are executed automatically when an instance or database is created. One might have to uncatalog a database explicitly before installing a new version of DB2 and cataloging it back after that. One might also want to recatalog the database to change its authentication type (client, server, etc.).

Cataloging remote databases for a client-server communication

- However, cataloging becomes absolutely necessary if one wants to set up a client to access a remote database (one might need to do that for various reasons including federated database configuration, replication, simply to be able to use and administer a database remotely, etc.): first, one must catalog remote DAS, second, one must catalog a remote instance, and finally one must catalog a remote database. After that one can access remote database as if it is a local database (analog of NFS mount).
- One can do cataloging either using SQL (see the next slide) or using db2cc.
- DB2 documentation is not clear about the need to catalog.

Cataloging remote databases for a client-server communication

To catalog the remote system (server):

CATALOG ADMIN TCPIP NODE parisn REMOTE paris.ligo-la.caltech.edu REMOTE_INSTANCE db2as SYSTEM parissystem OSTYPE linux

To catalog the remote instance from the remote system:

CATALOG TCPIP NODE ldasdbn REMOTE paris.ligo-la.caltech.edu SERVER 50002 REMOTE_INSTANCE ldasdb SYSTEM parissystem OSTYPE linux

To catalog the remote database from the remote instance:

CATALOG DATABASE test1 AS atest1 AT NODE ldasdbn

To list systems:

db2 => list admin node directory

Node Directory

Number of entries in the directory = 2

Node 1 entry:

Node name = GENERATE
Comment = Local workstation
Protocol = LOCAL
Instance name = db2as

Node 2 entry:

Node name = PARISN
Comment =
Protocol = TCPIP
Hostname = paris.ligo-la.caltech.edu
Service name = 523

To list remote instances:

db2 => list node directory

Node Directory

Number of entries in the directory = 1

Node 1 entry:

Node name = LDASDBN
Comment =
Protocol = TCPIP
Hostname = paris.ligo-la.caltech.edu
Service name = 50002

To list local and remote databases:

db2 => list database directory

```
 Number of entries in the directory = 4
.....
Database 2 entry:

 Database alias                 = ATEST1
 Database name                  = TEST1
 Node name                      = LDASDBN
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Remote
 Catalog node number            = -1
.....
Database 4 entry:

 Database alias                 = SAMPLE
 Database name                  = SAMPLE
 Local database directory       = /home/db2inst1
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Indirect
 Catalog node number            = 0
```

DB2 Discovery

- DB2 Discovery mechanism helps to catalog a remote server or database in db2cc without knowing details about the remote server/instance/database. This information is discovered automatically and a user is given choices.
- Servers indicate whether they support discovery by setting discover parameter of DAS to disable/known/search. Instances, databases on the server specify whether they want to be discovered by setting discover_inst/discover_db parameter to enable/disable.
- Clients set discover parameter in their instance to disable/known/search to indicate how they search for available remote servers/instances/databases.

DB2 Discovery

- If discover=search on a client, a client broadcasts a request to find DB2 servers that support 'search' discovery. As a result, db2cc gets a list of visible servers to catalog. For each cataloged server, db2cc also gets a list of instances visible for remote cataloging. For each remote instances cataloged, it gets a list of visible databases to catalog.
- The network might be too large to search exhaustively for visible DB2 servers. Therefore there is another discovery method called 'known'. Client that supports 'known' or 'search' discovery can specify a server that supports 'known' manually. After that the server would help the client to catalog visible instances and databases.
- Discovery only controls visibility, not accessibility!!! One can catalog objects whether they are visible or not.

DB2 Authentication

- Authentication is performed by an external to DB2 security facility that might be part of OS (on UNIX each instance runs in a separate account) or a separate product.
- How and where authentication is used is specified per server instance:
  - **SERVER Authentication takes place at the server by its OS when a userID and password are specified during an attempt to attach to an instance or connect to a database. This is default.**
  - **SERVER_ENCRYPT Same but all the passwords are encrypted at the client.**
  - **CLIENT First authentication takes place at a client. Next it might still happen at a server as well depending on some other parameters at a server that specify whether all the clients are trusted or not.**

DB2 Authentication

- **DCS** Similar to server. Used rather differently with DRDA Application Server. Used mostly with DB2 Connect.
- DCS_ENCRYPT
- DCE Authentication occurs at the server using DCE Security Services instead of OS.
- DCE_SERVER_ENCRYPT
- KERBEROS Use encrypted keys
- KERBEROS_SERVER_ENCRYPT If the client authentication type is not kerberos use SERVER_ENCRYPT.

- We should probably user SERVER or SERVER_ENCRYPT authentication. I tried SERVER and CLIENT.

Federated system

- Federated system is a server that supports queries that reference data located in different local or remote databases as if the corresponding tables were located in one database: for example, one can do joins or unions between table A from DB2 database on the remote server, table B from DB2 database located on the same machine as the federated system and table C from the remote Oracle database. DB2 supports federated system configuration.
- Location transparency: users refer data objects by nicknames and do not have to know where they are physically located when submitting a query to a federation system. Data can be moved, nicknames updated without any changes to applications

Federated system

- Federated system is just another client for data sources, it does not monopolize or restrict an access to the sources. Data sources do not need to know that they have been used in a federation system.
- Distributed queries are limited to read-only operations. Some DB2 specific utility operations are not supported: load, import, reorg, etc. LOBs are not supported.
- One can, however, use a pass-through facility to submit SQL statements directly to data sources in their native SQL dialect. One can also write to the source databases using pass-through access.

Federated system

- To configure the federated system, one must install it with 'Distributed Join' support enabled.
- If non-DB2 databases will be used as sources, DB2 Connect should be installed.
- Set the database manager configuration parameter 'federated' to 'yes'.
- Create wrappers. A wrapper loads a library used to access a particular class of data sources. For DB2 data source load choose DRDA wrapper.
- Create servers. A server describes a data sources: its type, location, wrapper to use, authorization information and other server options.

Federated system

- Create nicknames for tables, views, aliases for the objects in the data sources.
- You might also need to define user mapping, data type mapping, function mapping, index specifications (to improve performance of the distributed queries), etc.

Federated system: example

Two local databases: SAMPLE and FED1. SAMPLE contains table EMPLOYEE. FED1 contains table T1.

```
connect to fed1
create wrapper DRDA
catalog local node HERE instance db2inst1
catalog database sample as fsample at node HERE
create server LSERVER type db2/linux version 7.1 wrapper DRDA authorization
"db2inst1" password "****" options ( node 'HERE', dbname 'sample')
create nickname "FEMPLOYEE" for "LSERVER"."DB2INST1"."EMPLOYEE"
```

```
list tables
Table/View                      Schema          Type  Creation time
------------------------------- --------------- ----- --------------------------
femployee                       DB2INST1        N     2002-04-07-18.32.26.685849
T1                              DB2INST1        T     2002-04-07-17.40.20.057967
```

N - nickname, T - table. T1 is in fed1 database, employee is in sample database. Without federated system you could not access them in a single query. Now you can. Similarly it works with remote databases but you have to catalog the remote DAS and provide user mapping.

Replication in DB2

- DB2 provides replication facilities to maintain the same set of data in more than one locations. It works both between DB2 and some non-DB2 databases such as Oracle, Sybase, MS SQL Server.
- The configuration consists of
  - **Control tables** that can be stored anywhere: some database on a target or source servers, or even on some other computer, control server, whose database is cataloged on a target and source servers;
  - Change-capture mechanism: changes to the source database are captured from log files to temporary tables;
  - Apply program: reads data from these temporary tables on the source server and copies data to the target database.

Replication in DB2

- To configure replication, one first must set logretain parameter of the source database to 'capture' and, perhaps, increase the number of primary logs.
- Second, one must declare which tables on the source server are available for replication (define replication sources).
- Third, one creates a replication subscription:
  - maps a set of replication sources (tables, views) to targets,
  - defines for each source which columns and rows to replicate,
  - defines a type of a target table,

Replication in DB2

- specifies when a replication occurs (either all the members of the subscription set are replicated or none at each replication cycle):
  - Interval timing
  - Continuous timing
  - Event timing
  - On demand timing (supported only to Windows targets?)
- specifies which SQL statements to execute before or after each replication cycle (for example, one might want to clean various auxiliary tables).

- Forth, one starts caption program on the source server: *asnccp sample cold noprune* *The capture program examines the log files and captures changes made to the source since the last replication to temporary tables.*

Replication in DB2

- Fifth, one starts apply program: *asnapply deptqual copydb* . It usually runs on a target server but can run on any computer that can connect to source, target and control servers.
- During the first replication cycle the sources are fully copied to targets. During the consequent cycles only changes made to sources after the last replication cycle are copied to targets.
- Most replication configuration support LOBs.
- One can configure target tables that are joins or unions of existing source tables.

Replication in DB2

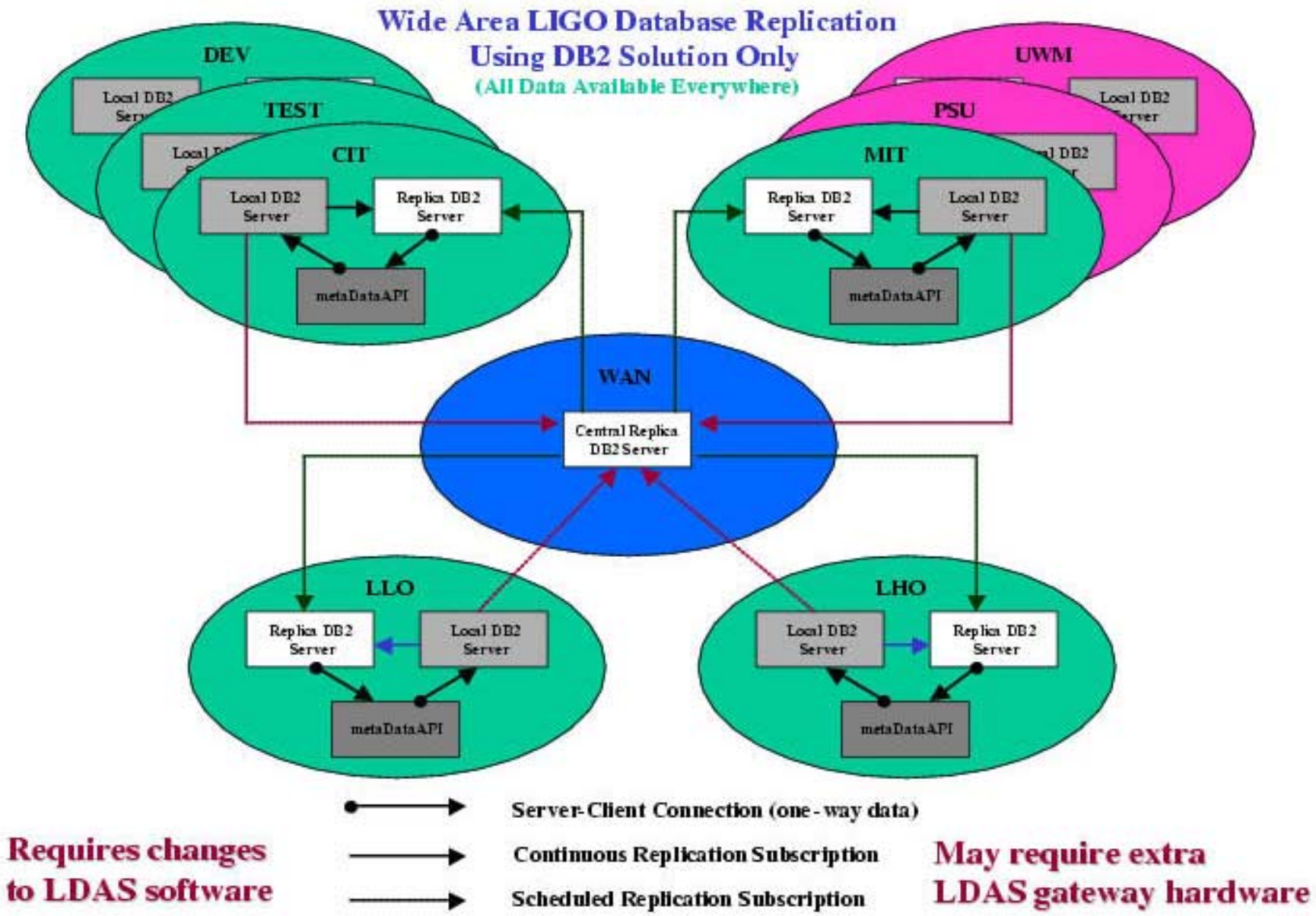
- Target table types:
  - User copy tables -- read-only copies of sources;
  - Point-in-time tables -- read-only copies of sources with an extra timestamp column that shows when the last update was made;
  - Aggregate tables -- read-only tables that use SQL column functions (sum, avg, etc.) to compute various summary information:
    - Base aggregate tables -- summary of source content
    - Change aggregate tables -- summary of changes to source content;
  - Consistent-change-data (CCD) tables -- contain data from committed transactions (many different types of CCD tables), can be used in between source and target to speed things up (e.g. source -- > CCD --> multiple targets);

Replication in DB2

- Replica tables -- the only target tables that are not read-only: applications can write to them, changes are propagated back to the source and from there to all other replica tables, replica do not support LOBs; replica requires conflict handling if, for example, source and target or two targets update the same row;
- User tables -- a source table for replica tables is a target table itself.

How to setup LDAS metadatabase replication

- Kent's big picture (next slide):
  - Separate network interfaces on metaservers or DB2 (DB2 Connect) on gateway?
  - Two metaservers at each site? Or two databases/instances on the same metaserver?
  - Changes to LDAS?


Wide Area LIGO Database Replication
Using DB2 Solution Only
(All Data Available Everywhere)
DEV
TEST
CIT
Local DB2 Server
Replica DB2 Server
metaDataAPI
UWM
PSU
MIT
WAN
Central Replica DB2 Server
LLO
LHO
Server-Client Connection (one-way data)
Continuous Replication Subscription
Scheduled Replication Subscription
Requires changes to LDAS software
May require extra LDAS gateway hardware

How to setup LDAS metadatabase replication

- First stage of implementation:
  - Required hardware: only central metaserver; network cards for site metaservers would be nice but not necessary;
  - If no network cards on metaservers, DB2 (or DB2 Connect) runs on gateway, catalogs databases from local and central metaservers, stores control tables, runs apply program;
  - No changes to LDAS: just replicate data on central server, LDAS does not use for now;
  - Use LDAS_TST or L[LH]O_TEST databases with the data generated artificially at approximately the same rate as during the engineering runs;
  - The goal of this stage is to get enough experience with DB2 replication mechanism to test if this is the way to go.

How to setup LDAS metadatabase replication

- Second stage:
  - Setup the joint database on sites metaservers (or second sites metaservers) and replicate data there from the central metaserver and from the first metaserver;
  - Again, artificial data is generated at the production rates;
  - The goal is to test if we have sufficient bandwidth.
- Third stage:
  - Implement the necessary changes to LDAS to utilize the joint metadatabase at sites.