Achieving the Holy Grail of Rolling Database Upgrades: User Perspective

Kenney Snell
DBA

September, 25 2008
• UPS was founded by Jim Casey in Seattle Washington in 1907
• UPS has become an expert in transportation, growing from a small messenger company to a leading provider of air, ocean, ground, and electronic services.
• 9th Largest Airline in the world
• UPS WorldportSM is the largest fully automated package handling facility in the world and serves as the airlines all-points international air hub. UPS serves nearly 800 airports worldwide.
Worldport℠, Louisville, Kentucky
January 25, 1988
- UPS Airlines is certified by the Federal Aviation Administration

February 1, 1988
- UPS Airlines operates its first flights
Objectives
Rolling Database Upgrade Process

Using the Transient Logical

- Test the upgrade process in a UPS lab / development environment
- Integrate and certify the upgrade process in a UPS “real world” application product test environment
- Document and add the upgrade process to our strategy for performing database upgrades
Agenda

Introduction

Business Case: Rolling Database Upgrade

Testing Environment: Rolling Database Upgrade (Transient)

Critical Tasks Failures / Fallback Options (Transient)

What we learned – UPS Application
Business Case – Rolling Database Upgrade

- Focus on High Availability and Disaster Recovery
  - Unplanned AND Planned outages
- Planned Downtime Service Level Agreement (SLA)
  - Maximum allowed downtime for any single event is 15 minutes
- Impossible to achieve using traditional methods when upgrading to Oracle 11g
- Rolling database upgrades achieve our SLA with time to spare
Testing Lab Environment

Equipment
- Compaq DL-585-G2 servers
- Red Hat Enterprise Linux 5.1
- HP Service Guard clustered database environment

Oracle Binaries
- Patchset Release Oracle 10.2.0.3 to 10.2.0.4 (64 bit)
- Full Release Oracle 10.2.0.4 upgraded to 11.1.0.6 (64 bit)

Databases
- Data Guard (A ➔ B)
- Streams (A ➔ E)

Applications
- Java 1.5 with Swingbench (Order entry application)
Testing Lab Environment

Primary Site

Application Server Tier

Application

Database Server Tier

Primary

Database: A
Version: X

Log Repository

Streams

Physical Standby

Database: B
Version: X

Log Repository

Streams

Alternate Site

Application Server Tier

Application

Database Server Tier

Physical Standby

Database: B
Version: X

Log Repository

Streams

Clients

Clients

Log Repository

Full Backups

Full Backups

REDO

Full Backups

Full Backups
# Critical Tasks Failures / Fallback Options Using Transient Logical

<table>
<thead>
<tr>
<th>Critical Tasks Failures</th>
<th>Fallback Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Conversion of the physical to logical (Database B) fails</td>
<td>Physical Standby Restore (Database B)</td>
</tr>
<tr>
<td>2. Upgrade logical Standby (Database B) fails</td>
<td>Downgrade or Flashback (Database B)</td>
</tr>
<tr>
<td>3. Testing Fails: application, data, database (Database B)</td>
<td>Physical Standby Restore (Database B)</td>
</tr>
<tr>
<td>4. A. Conversion of Logical to Physical fails (Database A)</td>
<td>Recreate Physical Standby from Backup</td>
</tr>
<tr>
<td>4. B. Upgrade physical Standby (Database A) fails</td>
<td>Recreate Physical Standby from Backup</td>
</tr>
<tr>
<td>5. Testing Fails: application, data, database (Database A)</td>
<td>Downgrade</td>
</tr>
</tbody>
</table>
3. Application Testing / Verification Fails (B)

1. Conversion (SQL Apply)
   - Clients
   - Version: X
   - Version: X

2. Upgrade B to X + 1
   - Clients
   - Version: X
   - Version: X +1

3. Run in Mixed Mode
   - Clients
   - Version: X +1
   - Version: X +1

4. Switchover to B and Upgrade A to X + 1
   - Clients
   - Version: X +1
   - Version: X +1

Failure

Retrieve Physical Standby from Backups

Full Backups
## Database, Site Failures / Fallback Options Using Transient Logical

<table>
<thead>
<tr>
<th>Critical Tasks Failures</th>
<th>Fallback Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Primary (Database A) – Database or Site Failure</td>
<td>Failover to Database B and use Log Repository D</td>
</tr>
<tr>
<td>4. Primary (Database B) – Database or Site Failure</td>
<td>Failover to Database A and use Log Repository C</td>
</tr>
</tbody>
</table>
2. Primary Site Fails During Upgrade of B

1. Conversion (SQL Apply)
   - S1: Version: X
   - S2: Version: X

2. Upgrade B to X + 1
   - S1: Version: X
   - S2: Version: X + 1

3. Run in Mixed Mode
   - S1: Version: X
   - S2: Version: X + 1

4. Switchover to B and Upgrade A to X + 1
   - S1: Version: X + 1
   - S2: Version: X + 1

Failover to B

Register Logs (D) into B

Recover B
UPS Application Architecture

Application

• Flight Operations and Scheduling
• Runs in two data centers (800+ miles apart)

Equipment

• PROLIANT DL585 G1 servers
• Red Hat Enterprise Linux 4.X 64 bit
• HP Service Guard clustered environment

Application Servers

• 4 BEA Servers, 64 bit (MQ / WLS 9.2)
• JDBC Multi Data Source
• Application (JVM)

Database Servers

• 4 Database Servers, Oracle 10.2.0.3 64 bit
• Data Guard (p100 → p100b)
• Streams (p100 → P200), P200 –DG→ P200b
What we learned – UPS Application

• Tested a Rolling Upgrade with the UPS Application
• Changed Application Database Connections
  • From BEA’s Multi Data Source to Oracle’s services
    ➢ Database Triggers (startup and role change),
    ➢ Database Service, and
    ➢ Connect string uses primary and standby database VIPs with a single service name
• Used same application servers during the database switchover
• Tested switchover with the application running (no need to stop or restart the application)
• Performed database upgrades in < 4 min. of application downtime
UPS Application / DG Switchover

Primary Data Center

Mainframe Systems

Queue

MQ

MCSGuard

BEA Application Server

Server #1

Server #2

Server #3
dr

Passive

Server #4
dr

Passive

Server #5

MCSGuard

Database Server

AIR

Triggers

- p100

Streams

- Reporting - p200

Alternate Data Center (DR)

Mainframe Systems

Queue

MQ

MCSGuard

BEA Application Server

Server #6

BEA Application Server

Server #7

BEA Application Server

Server #8

BEA Application Server

MCSGuard

Database Server

Service

AIR

Triggers

- p100

Streams

- Reporting DR - p200b

url>...

@(...) HOST= p100

... HOST = p100b

... Service = AIR </

url>...

@(...) HOST= p100

... HOST = p100b

... Service = AIR </
Data Guard and Streams - Rolling Upgrade

- Tested Configuration included a Streams Data Capture target
- Streams metadata has unsupported data types for SQL APPLY
  - Anydata, ROWID, AQ$_* and
  - Unsupported PL/SQL: DBMS_AQ
- Method: Transient Logical Standby
  - Updated Required Checkpoint SCN
  - Stopped Streams Capture prior to Rolling Upgrade
  - Register Archived Logs after converting Logical to Physical
    - RMAN> CATALOG RECOVERY AREA;
  - Restarted Streams Capture after Rolling Upgrade
Benefits: Rolling Database Upgrade

Proved the upgrade process works using a transient logical
  • Patchset Release - 10.2.0.3.0 to 10.2.0.4.0
  • Full Release - 10.2.0.4.0 to 11.1.0.6.0

Configured the UPS application for database re-connectivity
  • Streamlines the DG and business application switchover
  • Allows the business application to continue running which supports the SLA

Proved upgrade process works when Streams is part of the environment

Reduced application outage time to 4 minutes
  • Outage represents the DG Switchover
Benefits: Application Outage Savings

- Data Guard rolling database upgrades achieve our SLA with time to spare.

Total Application Downtime

- 10.2.0.3 to 10.2.0.4: Upgrade 15 minutes, Rolling Upgrade 4 minutes
- 10.2.0.4 to 11.1.0.6: Upgrade 55 minutes, Rolling Upgrade 4 minutes

Minutes

0 10 20 30 40 50 60

10.2.0.3 to 10.2.0.4 10.2.0.4 to 11.1.0.6

Upgrade  Rolling Upgrade

SLA