Db2 12 Continuous Delivery

New challenges for deployment

Roy Boxwell

SEGUS & SOFTWARE ENGINEERING

Session code: <V4>
Monday, October 2nd - 02:00 PM-03:00 PM

Platform: <Db2 z/OS>
Agenda

- Agile, Continuous Delivery, DevOps
- Db2 Code, Catalog, Function and Application Levels
- Activation/Deactivation of new code and how to fallback and when you can’t
- Different flavors of (pro-active) CD-screening and how it can be automated
  - Anomaly alerting based on Incompatibility Change Indicators (ICIs)
  - Dyn./Stat. access path change detection e.g. via Plan Management
  - Clone based code change pre-apply exploiting Backup System
  - Workload-KPI verification using SQL replay and KPI comparison
- Conclusion
Traditional vs Agile software development

• Traditional software development is a sequential process:
  • requirement definition – careful and complete
  • planning – visualization of the architecture
  • building – coding
  • testing – careful and complete
  • deployment – delivery and roll out

  → The key phases are the definition and planning, because it’s a hierarchical process.

• Agile software development is a diversified dynamic evolution:
  • adoption
  • coding
  • (constant) compatibility testing
Traditional vs Agile software development

• Both methodologies have their pro’s and con’s

• Traditional:
  • *Isolated*, serialized steps
  • Requires clear definition of requirements *in advance*
  • Less flexible when it comes to changes, fixes, enhancements
  • More solid with regards to quality
  • Stringent documentation and deployment
Traditional vs Agile software development

• Both methodologies have their pro’s and con’s

• Agile:
  • Less rigorous process
  • Smaller chunks, but highly interacting pieces
  • More (iterative) phases of revision and improvement
  • Requires a close collaboration between user and developer
  • Problems and improvements can be addressed easier and less disruptively
  • Time to market is usually shorter, but the overall effort may increase
Traditional vs Agile software development

• Both methodologies affect wide aspects of a business
  • User, Business Analyst and Developer have to collaborate differently depending on the chosen method
  • With Agile, the role of the developer changes due to the extensive interaction with the user
  • Agile is a more granular approach leading to more flexibility, but increases demands for interaction/compatibility of the chunks

• The decision is either or
  Either go for the traditional way of software development that deals with bulkier, unilateral systems with better ease and forgo the other advantages, or compartmentalize the large system into smaller agile developed processes and play a compatibility gamble.
Continuous Delivery and DevOps

Agile development requires near-time delivery

- Continuous Delivery (CD) is an approach to produce software in short cycles
- CD ensures that changes can be released at any time, considering building, testing and releasing faster and more frequent
- Key is a focus on more incremental updates
- CD requires a straightforward and repeatable deployment

Of course an enterprise can not just roll out a new code drop, but developers, operations, quality assurance, management and so forth have to work closer together.

DevOps (software DEVelopment and information technology OPerationS) aims to establish a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more reliably.
Agile, CD and our role in DevOPs

• Considering that Agile requires a close collaboration between development and the user, we are likely to be going closer to the labs – or are we already? Think about *IBM’s vote for enhancements*, or *Sponsor Users*.

• Considering that Agile doesn’t produce new versions, but an ongoing delivery stream, we are likely going to get new features, functions and fixes much quicker.

• Considering that Agile requires a total change in software development, we’ll likely see “radical internal changes” at IBM

• Considering that Agile and CD requires a different deployment methodology as well, we’ll very likely have to change our deployment.
Db2 levels

- Starting with Db2 12 GA, IBM is going forward to the CD model
- Only single phase migrations, no more CM, ENFM, NFM
- Enhancements may or may not require CATMAINT
- A new level is identified by
  - Version ID
  - Release number
  - Modification identifier
- New Enhancements and capabilities delivered as part of the maintenance stream as Function Levels
- Function Levels enabled via –ACTIVATE command
Db2 levels

- Db2 customers will see a significantly high volume of CD items in a single maintenance stream
- Entirely new releases, or versions will be very rare and only for major changes, like
  - Extending control structures
  - Enabling an architecture level set
  - Adopting a new compiler
- Recommended Service Upgrades (RSUs) are still available on a monthly/quarterly interval and may, or may not include new features
- Although a code drop may include fixes as well as new/changed features and functions, Db2 users keep control over what they activate – really?!!
Db2 levels

To be able to separate fixes from new/changed features, applying maintenance now consists of up to four different, hierarchical levels:

1. **Db2 code level** – Db2 libraries via SMP/E
2. **Db2 Catalog level** – Db2 Catalog changes via CATMAINT
3. **Db2 Function level** – general available functions via Db2 command (-ACTIVATE FUNCTION LEVEL)
   - Function levels may be skipped
4. **Application level** – individual application functions via Db2 BIND/REBIND (default via ZPARM: SQLLEVEL APPLCOMPAT, or BIND with APPLCOMPAT VvvRrMmmm)
5. **SQLLEVEL** – DSNHDECP for precompile or coprocessor (no more NEWFUN)
6. **SET CURRENT APPLCOMPAT** may be used for prior levels
7. **Client property** – clientApplCompat may be used for prior levels
Db2 levels

Db2 comes with a variety of sources to determine the levels

- The new Db2 Catalog table SYSIBM.SYSLEVELUPDATES maintains a history of the different levels
  - FUNCTION_LVL
  - PREV_FUNCTION_LVL
  - HIGH_FUNCTION_LVL
  - CATALOG_LVL
- ...and additional columns:
  - OPERATION_TYPE
    - “C” for catalog change – CATMAINT
    - “F” for function level – ACTIVATE
    - “M” for maintenance update
  - Timestamps and LRSN/RBA
Db2 levels

Db2 comes with a variety of sources to determine the levels

- Additional Catalog tables have a new column FUNCTION_LVL
  - SYSIBM.SYSDYNQRY
  - SYSIBM.SYSPACKCOPY
  - SYSIBM.SYSPACKAGE
  - SYSIBM.SYSQUERY
Db2 levels

Db2 comes with a variety of sources to determine the levels

- New Global Variables deliver further insights into the current levels
  - SYSIBM.PRODUCTID_EXT: pppvvrmmm
  - SYSIBM.CATALOG_LEVEL: VvvRrMmmm
  - SYSIBM.DEFAULT_SQLLEVEL (DECPSQLL): VvvRrMmmm
- Db2 Command -DISPLAY GROUP shows the different levels
  - CATALOG LEVEL
  - CURRENT FUNCTION LEVEL
  - HIGHEST ACTIVATED FUNCTION LEVEL
  - HIGHEST POSSIBILE FUNCTION LEVEL
- Db2 Command -DISPLAY LOCATION shows the Function Level of each connected system
Db2 levels

Db2 comes with a variety of sources to determine the levels

- RIBRELX – available via multiple interfaces after an applications “CONNECT”
  - The format was extended from VV.R.M to VV.R.MMM
- SQLCA and GET DIAGNOSTIC do not include the levels
  - SQLERRP: pppvrrrm : Note the missing mm
- CLI comes with a new infotype
  - SQLGetInfo API returns SQL_DBMS_FUNCTIONLVL showing the function level/buildlevel
- JCC introduces
  - getDatabaseFunctionalLevel() API driver returning the function level/buildlevel
- DDF Location Statistics shows the FL of connected system
- DDF accounting header show the FL of connected system
Activation of levels

1. SMP/E apply of PTFs, or RSUs may increase the Db2 code level – on the member level of a DS group
   → verify the system to be stable

2. CATMAINT (if needed) increases Db2 Catalog level
   → Code level fallback is not possible after CATMAINT

3. ACTIVATE FUNCTION LEVEL (if needed/available) increases the Db2 Function Level – on the group level of a DS group
   → new functions/behavior that aren’t related to SQL, DML, DCL available
   → REBINDs pick up Db2 Optimizer enhancements/changes
   → dyn. SQL (non-stabilized) affected accordingly

4. BIND/REBIND APPLCOMPAT and/or ZPARAM APPLCOMPAT (if needed/available) increases the Db2 Function Level of an individual package
   → new functions/behavior related to SQL, DML, DCL available
   → SET CURRENT APPLCOMPAT may be used for prior levels
   → clientApplCompat may be used for prior levels
Deactivation of levels

• Db2 Code Level fallback is only possible when no CATMAINT has been executed and the Function Level is still supported

• Db2 Catalog level fallback is impossible

• FUNCTION LEVEL fallback is possible via ACTIVATE FUNCTION LEVEL, but
  • the Db2 Code Level has to remain on the highest level

• BIND/REBIND APPLCOMPAT and/or ZPARM APPLCOMPAT can be used for Db2 Function Level fallback of an individual package
  → old functions/behavior related to SQL, DML, DCL

• SET CURRENT APPLCOMPAT may be used for prior levels

• clientApplCompat may be used for prior levels

• Unless you REBIND, packages bound with higher Function Levels remain functional...
Skipping levels

Db2 Function Levels may be skipped, but always include all new/changed features/behavior of prior Function Levels
CD Screening

- Now – is there a problem with Continuous Delivery?
- It depends ... it’s a matter of control & verification
  → Thus assure proper testing

APPLCOMPAT results before continuous delivery:
- The way some of the most popular scalar BiFs worked started changing with Db2 10 for z/OS. For example, CHAR:

```
SELECT CHAR(DEC_COL) FROM SYSIBM.SYSDUMMY1;
```

<table>
<thead>
<tr>
<th></th>
<th>CHAR(000.1)</th>
<th>CHAR(1000.)</th>
<th>CHAR(1.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>V10+</td>
<td>'.1'</td>
<td>'1000'</td>
<td>'1.1'</td>
</tr>
<tr>
<td>V9</td>
<td>'000.1'</td>
<td>'1000.'</td>
<td>'1.1'</td>
</tr>
</tbody>
</table>
CD-Screening

- Incompatible changes as of Db2 11:
  1: V9 version of CHAR(DEC) executed
  2: V9 version of VARCHAR(DEC) executed
  3: Unsupported character string representation of a TIMESTAMP
  4: V10 default SQL path used instead of V11
  7: SQLCODE -301 from a Db2 11 server
  8: Stored procedure data types
  9: TIMESTAMP TIMEZONE from DRDA
  10: V9 version of LTRIM, RTRIM, STRIP executed
  11: SELECT INTO with UNION
  1101: INSERT into XML column w/o XMLDOCUMENT function
  1102: XPATH evaluation resulted in error
  1103: Dynamic SQL ASUTIME limit RLF issue

- Additional issues:
  1104: CLIENT_ACCTNG longer than supported length pre-V11
  1105: CLIENT_APPLNAME longer than supported length pre-V11
  1106: CLIENT_USERID longer than supported length pre-V11
  1107: CLIENT_WORKSTNNAME longer than supported length pre-V11
  1108: CLIENT register longer than supported used for RLF
  1109: CAST(STRING AS TIMESTAMP) using invalid string lengths as of V11
  1110: Argument for SPACE function greater than 32764
  1111: Optional integer argument of VARCHAR greater than 32764
  1112: Empty XML element
CD-Screening

... and the first one announced for 12:

• 1201: changed SQLCODE for POWER results out of range

• Out-of-range result from POWER built-in function with DOUBLE arguments in Db2 12
  
  • Starting in Db2 12, when the POWER built-in function is invoked with DOUBLE arguments, and the result is out of range, warning SQLCODE +802 is returned, instead of error SQLCODE -802
  
  • Actions to take:
    In Db2 12, before function level 500 or higher is activated, identify applications with this incompatibility by starting a trace for IFCID 0376, and then running the applications. Review the trace output for incompatible changes with the identifier 1201. Adjust error processing to account for the change in the returned SQLCODE from an error to a warning
CD-Screening

The variations of levels and their dependencies/restrictions demand for an adjusted maintenance/migration

• We need to be aware of applications being affected by changed behavior
  • Since Db2 10 we can use IFCIDs 366* and 376 (Db2 11) to catch potential incompatibilities (Incompatibility Change Indicator – ICI)

• We need to be able to prescreen access path changes resulting from code changes
  • Since Db2 9 we can use PLAN MANAGEMENT to fallback to prior runtime structures
  • Since Db2 12 we have stabilized dynamic queries to freeze runtime structures
  • Since Db2 collections we have the opportunity to REBIND (distributed) packages into a separate collection to compare and switch
CD-Screening

The variations of levels and their dependencies/restrictions demand for an adjusted maintenance/migration strategy

- We need a reliable, solid and automated way to test the different levels
- The checks can’t take longer than a couple of hours, or a few days, otherwise we’ll never catch up
- We need to be flexible in choosing the scope of testing
  - ICI Detection
  - Access Path Precheck and Protection
  - Instant Cloning
  - Workload Capture/Replay
CDDC for Db2 z/OS
SOFTWARE ENGINEERING/SEGUS provides a powerful package that exploits years of experience in QA, AP Precheck, Cloning and Db2 Workload capturing

Continuous Delivery Deployment Check for Db2 z/OS
provides four levels of automated alerting and testing:

- ICI based anomaly alerting
- Static & Dynamic SQL Access Path Pre- and/or Post-Check
- Clone based code level checks exploiting Instant Copy (e.g. Flashcopy)
- Workload Capture/Replay with KPI verification

CDDC for Db2 z/OS let’s you decide the scope of testing, from incompatibility alerting up to a full system test including workload execution
ICI Detection - ICI based anomaly alerting:

*REBIND application static packages with higher APPLCOMPAT to exploit DDL/DML new functions/behaviors*

ICI Detection covers Function Level changes and pinpoints the origin of a statement (stmt id, program, package, collection, statement text)

- Incompatibility Change Indicators play a major role in Function/Application level handling
- Db2 itself tells us via IFCID 376 if potential incompatibilities exist, e.g.
  - Changes to the behavior of Built In Functions (BIFs)
  - Changes of SQLCODEs
  - Changes of SQL results
  - Deprecated support e.g. for TIMESTAMP formats
  - (...)
CD-Screening

ICI Detection - ICI based anomaly alerting:

<table>
<thead>
<tr>
<th>Package</th>
<th>Reason</th>
<th>WLX DB2 SSL</th>
<th>Statement ID</th>
<th>Section number</th>
<th>Plan name</th>
<th>Statement Timestamp</th>
<th>Transaction name</th>
<th>End User ID</th>
<th>Workstation name</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQADBACP</td>
<td>DE2 9 VARCHAR Usage</td>
<td>DC10</td>
<td>160,221</td>
<td>114</td>
<td>IQAP0610</td>
<td>2016-09-30-10.15.04.301036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQADBACP</td>
<td>DE2 9 VARCHAR Usage</td>
<td>DC10</td>
<td>2,361</td>
<td>11</td>
<td>IQAP0610</td>
<td>2016-10-11-10.14.304036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQADBACP</td>
<td>DE2 9 VARCHAR Usage</td>
<td>DC10</td>
<td>180,221</td>
<td>114</td>
<td>IQAP0610</td>
<td>2016-08-30-10.15.04.301036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQADBACP</td>
<td>DE2 9 VARCHAR Usage</td>
<td>DC10</td>
<td>180,221</td>
<td>114</td>
<td>IQAP0610</td>
<td>2016-10-11-10.12.304036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQADBACP</td>
<td>DE2 9 VARCHAR Usage</td>
<td>DC10</td>
<td>160,221</td>
<td>114</td>
<td>IQAP0610</td>
<td>2016-09-30-10.15.04.301036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQADBACP</td>
<td>DE2 9 VARCHAR Usage</td>
<td>DC10</td>
<td>1,260</td>
<td>11</td>
<td>IQAP0610</td>
<td>2017-01-25-09.42.51.304036</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Package: IQADBACP
- Reason: DE2 9 VARCHAR Usage
- WLX DB2 SSL: DC10
- Statement ID: 160,221, 2,361, 180,221, 180,221, 160,221, 1,260
- Section number: 114, 11, 114, 114, 114, 11
- Plan name: IQAP0610
- Transaction name: UNKNOWN, UNKNOWN, UNKNOWN, WLPVRST
- End User ID: UNKNOWN, UNKNOWN, UNKNOWN
- Workstation name: DB2CALL
CD-Screening
Access Path Check - Static & Dynamic SQL Access Path Pre- and/or Post-Check:

REBIND of packages with any APPLCOMPAT would pick up optimizer enhancements. Non-stabilized dynamic SQL would pick up optimizer enhancements.

➔ Access Path check covers Function Level Changes for both static AND dynamic SQL.
➔ Access Path check runs proactive (instead of PLANMGMT) and can be executed in your production environment (without harming applications), or in a virtualized test environment (simulates all optimizer relevant factors.)
➔ Access Path check clearly categorizes improvements and degradations (instead of PLAN STABILITY).
   ➔ Access Path check opens up improvements and suppresses degradations.
CD-Screening

Access Path Check - Static & Dynamic SQL Access Path Pre- and/or Post-Check:

Plan Table
Db2I V12R1M500

Comparison

Impact-Analysis
improved or equal

Worsened

Report

New Access Path Pattern

New FL behaviour

Plan Table
Db2P V12R1M100
CD-Screening
Access Path Check - Static & Dynamic SQL
Access Path Pre- and/or Post-Check:

<table>
<thead>
<tr>
<th>STMT No.</th>
<th>Section Number</th>
<th>DSC STMT ID</th>
<th>Impact</th>
<th>Serviceunits Old</th>
<th>Serviceunits New</th>
<th>Milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>2547</td>
<td>13</td>
<td>2547</td>
<td>DEGRADED</td>
<td>85</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>796</td>
<td>2</td>
<td>796</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>804</td>
<td>3</td>
<td>804</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>812</td>
<td>4</td>
<td>812</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>820</td>
<td>5</td>
<td>820</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>671</td>
<td>2</td>
<td>671</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>679</td>
<td>3</td>
<td>679</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>687</td>
<td>4</td>
<td>687</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>695</td>
<td>5</td>
<td>695</td>
<td>DEGRADED</td>
<td>152</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>DEGRADED</td>
<td>130</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>DEGRADED</td>
<td>95</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>DEGRADED</td>
<td>82</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>DEGRADED</td>
<td>149</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>968</td>
<td>7</td>
<td>968</td>
<td>IMPROVED</td>
<td>171</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1157</td>
<td>9</td>
<td>1157</td>
<td>IMPROVED</td>
<td>174</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>1</td>
<td>235</td>
<td>IMPROVED</td>
<td>178</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1194</td>
<td>2</td>
<td>1194</td>
<td>IMPROVED</td>
<td>150</td>
<td>131</td>
<td></td>
</tr>
<tr>
<td>2409</td>
<td>7</td>
<td>2409</td>
<td>IMPROVED</td>
<td>21</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>1194</td>
<td>2</td>
<td>1194</td>
<td>IMPROVED</td>
<td>175</td>
<td>191</td>
<td></td>
</tr>
</tbody>
</table>
CD-Screening

Instant Cloning - Clone based code level checks:

PTFs (RSUs...) are applied that may increase the Code Level (CL) of a Db2 system. After system is stable on maintenance, execute (If Any) catmaint.

Instant Cloning covers Code Level, Catalog Level and Function Level testing and allows the most comprehensive scope.

• For a full check, an entire Db2 system can be cloned in minutes. Object level cloning allows being more granular.
• FTP support transfers required data to different LPARs.
• Scheduling Environment and Routing support handles system setups with different environments.
• The entire process is guided and controlled.
CD-Screening

- Instant Cloning - Clone based code level checks:
- Scope of Cloning: Subsystem level

- Fully exploiting instant copy technology (e.g. Flashcopy)
- Supports DS↔NDS, as well as cross-version cloning
- Highly customizable and fully automated via XML scenario scheme
CD-Screening

- Instant Cloning - Clone based code level checks:
- Scope of Cloning: Object level

- Fully exploiting instant copy technology (e.g. Flashcopy)
- Supports DDL and/or data cloning
- Flexible include/exclude of dependent objects
- Powerful renaming capabilities
CD-Screening

Instant Cloning - Clone based code level checks:
CD-Screening

Workload Capture/Replay with KPI verification:

*After Function Level is considered stable – allow new application feature rollout.*

→ Workload Capture/Replay adds Application Level testing and completes the most comprehensive scope by executing sets of captured workload.

- Highly efficient IFCID (OPx) based capturing technology catches SQL for automated execution in the isolated cloned environment.
- Workload sets can be saved to represent quarters end, years end and other specific workload patterns.
- Tested workload is compared on a KPI level (e.g. # of getpages, rows returned, rows processed…) to report anomalies only.
Workload Capture/Replay with KPI verification:

- Workload sets stored in a Workload Warehouse Repository
- Pre-filtered to remove any unwanted/unsupported SQL
- Transferred to the clone environment
- Executed and KPI compared to automatically find anomalies
# CD-Screening

## Workload Capture/Replay with KPI verification:

<table>
<thead>
<tr>
<th>Selection 1</th>
<th></th>
<th>Selection 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Authorization ID</td>
<td>Package</td>
<td>Collection ID</td>
<td>Number of Statements</td>
</tr>
<tr>
<td>n/a</td>
<td>CMEMAPP</td>
<td>PTFCCOLL003</td>
<td>1</td>
</tr>
<tr>
<td>n/a</td>
<td>COSEAR</td>
<td>PTFCCOLL003</td>
<td>1</td>
</tr>
<tr>
<td>n/a</td>
<td>COQAPTF</td>
<td>PTFCCOLL003</td>
<td>1</td>
</tr>
<tr>
<td>n/a</td>
<td>DSNRER2</td>
<td>DSNTER2</td>
<td>3</td>
</tr>
<tr>
<td>n/a</td>
<td>DSNRER2</td>
<td>DSNREX</td>
<td>1</td>
</tr>
<tr>
<td>n/a</td>
<td>DSNTRAP</td>
<td>DSNTRAP</td>
<td>2</td>
</tr>
<tr>
<td>n/a</td>
<td>FILPROD</td>
<td>PTFCCOLL003</td>
<td>1</td>
</tr>
<tr>
<td>n/a</td>
<td>IADMPTF</td>
<td>PTFCCOLL003</td>
<td>1</td>
</tr>
<tr>
<td>n/a</td>
<td>IMEMBI</td>
<td>PTFCCOLL003</td>
<td>1</td>
</tr>
</tbody>
</table>

### Selection details

<table>
<thead>
<tr>
<th>Column name</th>
<th>Selection value 1</th>
<th>Selection value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest CPU Time</td>
<td>0.062298</td>
<td>0.062298</td>
</tr>
<tr>
<td>Sum of Elapsed Time</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Average Elapsed Time</td>
<td>0.002149</td>
<td>0.002149</td>
</tr>
<tr>
<td>Highest Elapsed Time</td>
<td>0.079543</td>
<td>0.079543</td>
</tr>
<tr>
<td>Sum of Executions</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>Sum of GETPAGES</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>Average GETPAGES</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Highest GETPAGES</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>Sum of Synchronous Buffer Reads</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Sum of Synchronous Buffer Writes</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>
CD-Screening

All tests can be combined and included in an automated execution stream to make the entire testing of a new Db2 code drop a standardized process.

A typical setup is for example:

- Have ICI Detection continuously govern the production system (overhead minimal)
- Make Access Path Check a vital part of REBIND processing as well as DSC check to keep control of dynamic SQL
- Before PTF/RSU apply:
  1. Clone into an isolated environment for CD-screening, run Workload Replay
  2. Apply the PTF/RSU and if needed CATMAINT
  3. Run Workload Replay
  4. Check system behavior and KPI report for anomalies
# CD-Screening

The scope and the environment can be chosen flexibly

<table>
<thead>
<tr>
<th></th>
<th>Code Level</th>
<th>Catalog Level</th>
<th>Function Level</th>
<th>Application Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICI Detection</td>
<td>X</td>
<td>X</td>
<td>Test and/or Production</td>
<td>Test and/or Production</td>
</tr>
<tr>
<td>AP Check</td>
<td>X</td>
<td>X</td>
<td>Test(^+) and/or Production</td>
<td>Test(^+) and/or Production</td>
</tr>
<tr>
<td>Instant Cloning</td>
<td>Test</td>
<td>Test</td>
<td>Test</td>
<td>Test</td>
</tr>
<tr>
<td>Capture/Replay</td>
<td>Test</td>
<td>Test</td>
<td>Test</td>
<td>Test</td>
</tr>
</tbody>
</table>
Conclusion

- Compared to prior versions the path into the Agile world and their influence on existing behavior/compatibility started already in Db2 10 with BIF Compatibility

- APPLCOMPAT was first introduced in Db2 11

- The traditional three yearly release cycle doesn’t fit to the technological development these days

- Retrofitting enhancements burns too many resources

- The idea of Function Levels should make the continuous delivery stream manageable
Conclusion

IBM is investing in new functionality with the key variable TIME, we must invest in TEST if we want to keep our level of stability.

• Typically customers spent months to test new code, but that won’t fit the continuous delivery stream.

• A multilevel, highly automated test framework is a reliable, robust solution to exploit and benefit from the new directions.
Recommendations

Understand Continuous Delivery

• Get familiar with the various levels and what they affect
• Verify and adjust your migration procedures
• Check the changes that affect a migration
• Resolve application incompatibilities before setting APPLCOMPAT to a new level
• Use the TEST option before activating a new level

Further information:

• Redbook: IBM Db2 12 for z/OS Technical Overview
• IBM Knowledge Center
Conclusion

Continuous Delivery Deployment Check for Db2 z/OS

- Comes with multicomponent quality assurance
- Depending on the importance of the system to maintain, tests can be extended, or reduced
- A focus on automation helps to test complex Db2 applications
- Exploiting efficient storage technology keep costs low
- Automated cloning scenarios clone an entire production system in minutes, ready for testing
- Workload Capture/Replay combined with KPI comparison assures that changes in Db2 behavior are detected quickly and efficiently
BIF/ICI HealthCheck for Db2 z/OS

• A handy batch job, or started task that reports BIF incompatibilities in Db2 10, Db2 11, or Db2 12

• Both IFCID 366 and 376 can be captured as long as desired

• At the end of the capture process, or during -if desired- a report shows any affected dynamic, or static statement along with it’s
  • PLAN
  • COLLECTION
  • PACKAGE
  • Number of occurrences (366)
BIF/ICI HealthCheck for Db2 z/OS

BIF Health Report for group GSA12A13 started at 2015-09-24 15:15:58.48

<table>
<thead>
<tr>
<th>Type</th>
<th>Plan</th>
<th>Collection</th>
<th>Package</th>
<th>ICI Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyn</td>
<td>DSNESPUR</td>
<td>DSNESPUR</td>
<td>DSNESM68</td>
<td>1 5</td>
</tr>
<tr>
<td>Dyn</td>
<td>DSNESPUR</td>
<td>DSNESPUR</td>
<td>DSNESM68</td>
<td>5 4</td>
</tr>
<tr>
<td>Stat</td>
<td>BIFTEST</td>
<td>BIFTEST_TEST</td>
<td>O2TESTB</td>
<td>1 2</td>
</tr>
</tbody>
</table>

BIF Health Report for subsystem DB10 started at 2015-09-25 10:48:17.23

<table>
<thead>
<tr>
<th>Type</th>
<th>Plan</th>
<th>Collection</th>
<th>Package</th>
<th>ICI Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyn</td>
<td>DSNESPUR</td>
<td>DSNESPUR</td>
<td>DSNESM68</td>
<td>1 2</td>
</tr>
<tr>
<td>Stat</td>
<td>BIFTEST</td>
<td>BIFTEST_TEST</td>
<td>O2TESTB</td>
<td>1 2</td>
</tr>
</tbody>
</table>
BIF/ICI HealthCheck for Db2 z/OS

Download* the Freeware at https://www.segus.com/bif

*requires a free account at segus.com
Roy Boxwell
SOFTWARE ENGINEERING
r.boxwell@seg.de

Session code: <V4>

Db2 12 Continuous Delivery –
New challenges for
deployment

Please fill out your session evaluation before leaving!