1 Scan Fähigkeit

Die Applikation unterstützt den Scan nach 3 unterschiedlichen Regel Sets, die auch unterschiedliche Ansichten für die zeitlichen Aspekte einer Migration beinhalten. Alle Regel Sets können natürlich erweitert werden, sollten aber hinsichtlich der Hauptfunktionalitäten

- Cluster Scan,
- Unix Scan
- Userdefined Scan

nicht gemischt werden. Soll eine Domäne nach allen drei Kriterien analysiert werden, sollten auch 3 unterschiedliche Scans mit drei unterschiedlichen Datenbanken durchgeführt werden.

Man hat dann zwar zwei Result-Sets, die auch beide abgearbeitet werden müssen. Aber es ist klar ersichtlich, welche Applikation in spezifische Klassen fallen.

Zudem können dadurch bessere Ergebnisse bei der Zeitabschätzung geliefert werden.

1.1 Versionsthematik:

Die aktuell ausgelieferten Regel Sets werden bei Erscheinen neurer Domino Versionen natürlich immer sorgfältig untersucht, ob sie nicht erweitert werden müssen. Aber gerade bei der Unix Analyse und einer User-definierten Suche z.B. nach feststehenden Begriffen wie Servernamen, gibt es kaum Domino Versionsabhängigkeiten. So ist die Suche nach direkten Dateizugriffen immer gleich, es kommen unter Umständen nur weitere Befehle hinzu.

Bei der Untersuchung der Clusterfähigkeit hat sich seit R4 nichts geändert. Der einzige Befehl, der von LotusScript Seite in der Lage ist, eine Datenbank Failoversicher zu öffnen, ist NotesDatabase.OpenWithFailover(...).
Genauso trifft z.b. für @DBLookup, @DBColunm, @Picklist, etc zu, dass sie nicht mehr in einer Failover (oder Loadbalanced) Situation arbeiten, wenn der Server auf den verwiesen wird, ein fester String ist oder aus einem Profile kommt, wohingegen die Arbeit mit @DBName sehr wohl arbeitet.

Gerade der Umzug von Applikationen von einem nicht geclustertem Umfeld in ein geclustertes wirft viele dieser Fragen auf. Prinzipiell laufen die Applikationen weiter, Probleme gibt es nur bei Serveragenten (Event und zeitgetriggert) und im Frontend bei einem aktiven Failover oder bei eingesetztem Loadbalancing.
Da die Applikationen meist nicht mit der Prämisse "Applikationscluster sicher" entwickelt wurden, sind hier die meisten Änderungen zu befürchten und diese sind dann auch noch semantisch abhängig. So würde z.B. der Lookup

- @DBLookup(""; @subset(@dbname; 1):"test.nsf"; "LookupView"; "1234"; 1)) ist failoversicher arbeiten, wogegen dieser Lookup
- @DBLookup(""; @GetProfileField("setup"; "Servername"):"test.nsf"; "LookupView"; "1234"; 1)) in einer Failover Situation (Loadbalancing) nicht mehr arbeiten würde.

1.2 Zeiten

Die Scan-Zeiten sind stark abhängig von der Leistungsfähigkeit des Clients und der Komplexität der zu untersuchenden Applikationen. Man kann davon ausgehen, das der Scanner im Schnitt 1 Minute pro Datenbank braucht, was im Endeffekt bei 1400 Applikationen auf 24h hinausläuft.

2 Zeitliche Bewertung

Alle gefundenen kritischen Codesequenzen werden abhängig von der Suchart unterschiedlich bewertet:

2.1 Clusterfähigkeit:

Hier werden alle kritischen Kodesequenzen, die Existenz von Autor und Leserfeldern und vor allem, was zusätzlich als Konstante mit relativ großem Einfluss eingeht, die Anzahl von getriggerten Serveragenten (Keine Web Agenten!) zeitlich bewertet.

Letzterer Konstante wird deshalb soviel Zeit zugemessen, da an dieser Stelle unter Umständen ein hoher Aufwand betrieben werden muss, um semantische Änderungen des Codes erforderlich werden (Auf welchem Server wird der Agent betrieben, muss er Clusterfähig sein, ...

Anzahl gefundener Elemente, die kritische Codesequenzen enthalten:
Pro Element (Agent, Action (View, Maske, Hotspot, ….) * 30 Minuten
Sind Autor oder Lesefelder vorhanden:
+ 30 Minuten
Pro Event- oder zeitlich getriggerten Server- Agenten
Anzahl * 120 Minuten

Formel :
CriticalCodeCount * 30 + 30 (exist(Author | Reader)) + ServerAgentCount * 120

2.2 Unix Fähigkeit

Nur zeitlich bewertet, wenn kritischer Code der Klassifizierung F (Code, der auf dem Server ausgeführt wird) gefunden wurde. Dann werden die Anzahl kritischer Code Elemente (LSX, direkte Dateizugriffe, COM, ...) mit 60 Minuten multipliziert. Um dazu noch eine generelle Konstante zu bringen, wurde die Anzahl der Serveragenten /Web und getriggert) mit jeweils 30 Minuten dazu addiert.

Formel:
(CriticalCodeCountCLassF + (ServerAgentCount + WebAgentCount) / 2)) * 60

2.3 User definiert

Diese Sektion kann speziell nach dem Bedarf des Kunden angepasst werden. Aktuell wird die Berechnung auf die Webfähigkeit bzw. den Änderungsaufwand, der dazu führen würde, analysiert.

In anderen Fällen könnte hier die Suche nach fest vorgegebenen Organisationseinheiten stehen, die sich bei einer Rezertifizierung ändern und fest im Code vorkommen.

3 Vorkonfigurierte Regel Sets

Diese Sets und die Suche selbst werden in einem Profildokument festgelegt. Anhand der hier getroffenen Angaben und Suchbegriffe findet eine Einstufung der Applikation in entsprechende Klassifizierungen statt, die Sie der Applikationsbeschreibung entnehmen können.

3.1 Regel Set, nach dem bei der Cluster Analyse gesucht wird.

getDatabase(<>*("",*
OpenByReplicaID(<>*("",*
.OpenDatabase(<>*("",*
.Open(<>*("",*
new NotesDatabase(<>*("",*
@DBLookup(":*")*<>*@DBName*><*("":"":""*><*("":"":""*><*(*Cache":""*
@DBColumn(":*")*<>*@DBName*><*("":"":""*><*("":"":""*><*(*Cache":""*
@PickList(":*")*<>*[]NAME]*><*[]ROOM]*><*[]RESOURCE]*
@RefreshECL(":*")*<>*@DBName*><*(*];"":*
@EditECL(":*")*<>*@DBName*><*(*];"":*
@[]FileOpen*]*:*<>*@DBName*><*(*];"":*
@[]Compose]*:*)*<>*@DBName*><*(*];"":*
@[]RenameDatabase]*:*)*<>*@DBName*><*(*];"":*
@[]AddDatabase]*:*)*<>*@DBName*><*(*];"":*
@[]OpenHelpDocument]*:*)*<>*@DBName*><*(*];"":*

3.2 Regel Set, nach dem bei der Unix Analyse gesucht wird

*Open * As *
Close
Unlock *
Lock *
Reset
*=*FreeFile *
*dir(*)
*ChDrive *
CurDir("")<>*CurDir("")
*setFileAttr *,*
getFileAttr(*)
:
"[A-Za-z]:"
date=
time=
Declare * Lib *
("ODBC"
*uselsx *
*ActivateApp *
CreateObject(*)
GetObject(*)
isObject(*)
isUnknown(*)

3.3 Regelsnytax

Wobei die Syntax der Regeldefinition keine Standart Regular-Expressions darstellen, sondern eine Mischung aus Notes Regular-Expressions und einer Erweiterung der gleichen für Variationen darstellt, die folgender Syntax unterworfen ist:

> Here we define the patterns for searching special code, which is responsible for classifying the database to D (critical code).
>
> Each entry will be a regular expression (LotusScript syntax) for searching such code. You could divide an entry in an include part, which means, that we first look for code, which match the pattern and in an exclude part, which reject the same code, if we match such a pattern. In the exclude part, you could define more than one entry, and we look for each entry one after another. So all entries in the exclude sections represents an ored operation.
>
> The syntax is:
>
> include pattern[<>[exludePattern][><exludePattern]]
>
> _____
>
> LotusScript regular expressions:
>
> A string expression that can include any individual ANSI characters and any of the wildcard characters or collections that follow. You can use as many wildcards as you need within a single patternString.
> Wildcard Matches
> ? Any one character
> # Any one digit from 0 through 9
> * Any number of characters (zero or more)
> [characters] Any one of the characters in the list or range specified here
> [!characters] Any one character not included in the list or range of characters specified here
>
> Matching characters in a list
>
> Enclose the characters between square brackets with no spaces or other delimiters between characters (unless you want the space character to be part of the list).
>
> Matching special characters
>
> To match one of these characters, include it in a characters list:
> Hyphen (-)
> Question mark (?)
> Asterisk (*)
> Number sign (#)
> Open bracket ([ )
>
> To match one of these characters, place the character anywhere within your wildcard specification except in a characters list or range:
> Comma (,)
> Close bracket ( ])
> Exclamation mark (!)
>
> Be sure to place the hyphen at the beginning of the list; if you're using the [!characters] format, the hyphen immediately follows the exclamation point, as in [!-123].