Redesign of Whitepaper Database & Java Code

Technical draft without details | Version 1.0 | Date: 28th April 2008

Content

Segmentation of existing tables in logic structure

Introduction

This document contains the first instructions, how to reorganize existing table structure – it describes required processes and outputs related to implementation.

Tables, fields contained in them and relations described in this document should be in this context considered as concept, not a strict guideline. Very well stipulated is the pure segregation of configuration, master and operation data.

General requirements on tables

Multi-mandate capability

All tables will be administered via column MANDANT and FIRMA with multi-mandate capability. The mechanism „Multi-mandant administering“ controls, which tables shall be included, respectively who is the owner of mandate and who is just participator.

Skill concept

Skills administration enables to execute comprehensive Recherché, respectively analysis according to data-mining principle.

Locking database

Based on the concepts dboptimistic (no active locks, but modifications must be confirmed before saving) and dbpessimistic (locking at the beginning of dialogue) shall be implemented as follows:

- pessimistic locking is not reasonable in system oriented on dialogue/transactions
- dboptimistic implies, that each data set is checked before saving using „Row version“ to find eventual changes. If a change was found, user is informed about this fact in dialogue listing the changes. Active user can decide, if the data should be merged, or replaced.

Optimization of data administration

Dimensions, metrics and data cubes will be defined based on database modellation and identification requests on database.

Optimization of database access

A database model includes number of tables, which are partially abstract, because the table is based on repeating domain entities. The abstract data set will be generated on entities' data set first after use of views.

Stored Procedures to execute necessary CRUD function will be implemented to enable access to tables/views. Furthermore, complete database-related application's logic will be also generated in form of Stored Procedures. Complete Stored Procedures will be implemented in .NET, or more exactly C#.

Implementation

To meet estimations of complexity of system core realization, reasonable statements may be formulated prior realization based on some analysis. The analysis is focused solely on *Application* and *Domain* levels establishing non-rigorous 4-level architecture.

Generally, the domain logic will be defined/formulated after the domain analysis in frame of functional specification. On higher level the business logic will be sophisticated and split in two parts:

- Application's logic – Semantics of application's logic corresponds with workflows and therefore is inherently stateless.
- Domain logic – Entities, values, rules, etc. and their interactions will be derived from domain logic. This part of business logic has defined status.

According to this logic-oriented view on implementation, another view on NEXA is necessary to be able to reach estimated complexity of implementation (and of course

previous software draft).

Based on context diagram it is possible to analyze, how the system will interact with specific environment:

- Interactive – user works with system.
- Batch-oriented – Data intensive calculations, migrations etc. will be executed in regular time intervals. These batch runs may be realized very quickly and resource-efficient.
- Event-based – Communication with other system initiates use of non-functional requirements, such as defect tolerance, performance, etc.

“What” and “How” are two dimensions, which are orthogonal one to each other and therefore significantly increase complexity of use. Both dimensions bring non-functional requirements, which partially exclude each other. The degree of complexity of developed application is very high even without non-functional requirement *Clustering*.

Clustering

Clustering is a central requirement on application. The goal of clustering is to implement load distribution and high availability. This goal will be achieved, if the application in case of redundant distributing application across number of nodes. There are more applications possible for clustering in JEE context. These applications enable to execute necessary synchronization of nodes using various methods.

Each alternative of clustering technology must be analyzed in regard to defined use and following aspects:

- Cluster implementation
- Failover service of cluster and its components
- Failover of HttpSession
- „Single Point Of Failure“ of Cluster topology
- Flexibility of topology layouts
- Serviceability

Distribution

Type of distribution is fundamental for serviceability and extensibility of system. New nodes will be added for cluster, if required. Existing nodes must be updated during maintenance on up-to-date development status.

There are two basic approaches to distribution of cluster nodes. Both of them have advantages and disadvantages:

- Shared Nothing – All cluster nodes get local installation of application. Advantage is that nodes are independent on each other. But there is a disadvantage - in case of implementation of new version of application, each node must be updated individually.
- Shared Disk – No local installation is performed on cluster nodes. Application is installed on network drive, connected to all nodes. Installation of application is easily serviceable. Disadvantage of this approach is Single Point Of Failure.

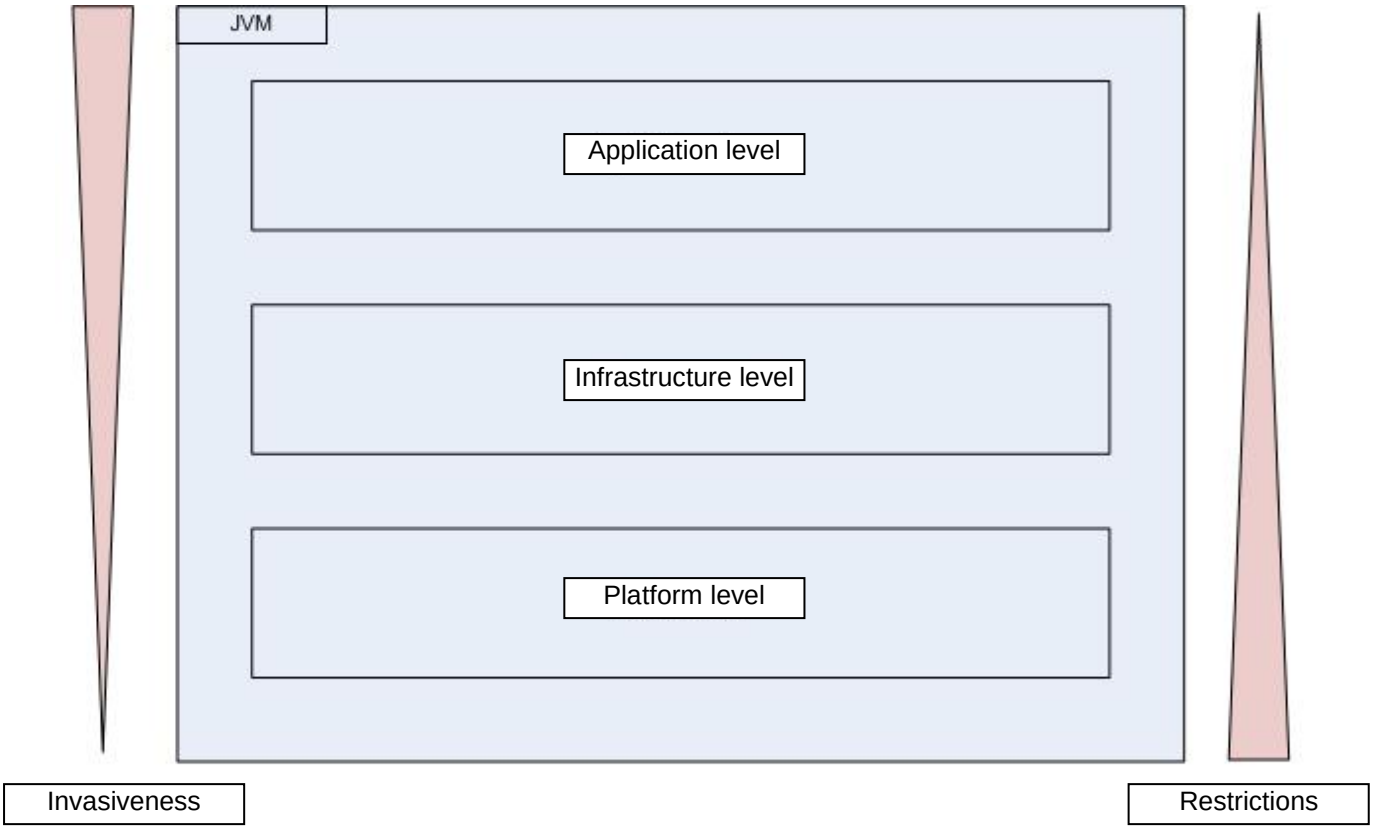
Doubtlessly elegant solution is a Shared Disk approach. It can be improved by using SAN instead of usual file server. It brings high availability of distribution of application.

Versions

Still the clustering of application is for software drafts, and especially for later implementations a challenge, which should never be underestimated. Clustering is based on redundant distribution of application, where the replicates must be synchronized one to another. Existing approaches and technologies precede each other diversely, where first must be answered the question, on which place within system stack is realized the necessary synchronization.

- Implementation of clustering within application on base of RMI/JMS, respectively proprietary development of Clustering approach - **application's level**
- Use of clustering library, such as Jgroups - **infrastructure level**
- Use of container realizing the clustering, such as EJB container according to JEE specification (Jboss uses for this for example Jgroups) - **platform level**
- Clustering of JVM – **except system stacks**

If we analyze the first 3 alternatives, it will be found out, that both criteria of **invasiveness** and **restrictions** have opposing effect.



As higher in system stack you are, as more invasive is the realization of clustering, because correspondent mechanism leaves too many traces in code. A container accepts clustering on the very bottom of stack. This forces the designer to observe predefined component model (such as EJBs).

Therefore the component model on the lowest level requires the highest restrictions defined for example in JEE specification (threading, File-IO, etc.). If you scroll in stack back on the top, restrictions disappear, because the designer must define, what, when, and how shall be synchronized.

Implementation of own clustering solution is not option, which could be realized, because the expenses, costs and risk would be exorbitant. The best option is to use the clustering library, even it is still very invasive. Use of EJB container is undesirable for customers and so it is no option.

Clustering JVM

There is also possibility to make clustering of JMV. Advantage of this approach is , that all synchronization activities are realized declaratively, and so the solution is not invasive.

However, some JMV classes are not suitable for clustering. This affects mainly the classes representing local resources. It is also important, that the semantics of JMV structure remains unchanged. Constructs, such as

- Static fields
- Threads
- Transactions
- Monitors (synchronized)
- Object identity
- Garbage Collector

etc. must during corresponding declarations remain semantically correct.

Cluster awareness within application

Except frame conditions implied by clustering approach, it is necessary to process further deliberations, to be able to distribute application across number of nodes.

In short, two aspect of application have been mentioned in this relation:

Considering code of application, it is possible to split the codes on internal and external codes. Internal code is a result of own design job of involved software engineer, external code is represented by libraries integrated usually in form of JAR. In course of adaptation of code providing control in form of resource for use in clustering environment, it is not possible to estimate for third-party libraries, if they are suitable for clustered use.

Another aspect is the status playing a role in context of application. There is a user-specific and application-specific status.

User-specific status arises in relation to type of client – administered either by client, or by server. Desktop application may administer user-specific status. But this is not possible in case of using a browser for purposes of web application. Because HTTP is a stateless protocol, the task of status management takes over the server in form of HttpSession.

The application has also a status, which may be assigned to particular group. When the domain status may be fully stored in database, is the domain-independent status fully stored in main memory.

An important part of selecting clustering technology is to answer the question, if the synchronization and failover for HttpSession and domain-independent status will be executed.

Measures and preparations for clustering applications

Following measures must be performed prior startup of realization of application:

- Estimation of expenses, costs and risks for use of clustering technology. Consider restrictions (such as component model EJB), impacts of modified semantics of linguistic constructs, eventually acceptation of known invasiveness in source code
- Analysis and selection of applicable clustering technology
- Realization of study in terms of "Proof Of Concept"
- Working out concept for testing according to clustered scenarios -> obvious additional expenses on testing related to application

Because today there is no established method for clustering Java application, it is possible to consider in frame of NEXA development definition of two partial projects for purposes of quality assurance:

- Development of analysis tool, which will verify the source code by means of static code analysis to meet the statements if the application clustering is possible. As extension, analysis of related third-party products shall be possible. In second step the anti-patterns shall be identified and correspondent refactoring must be formulated ant implemented.
- Development of concept, how to test the application in clustering context. Based on

this the test framework will be realized (eventually as extension to facto-Standard JUnit).