Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals of DAAAM International

IMPROVING APPLICATION PERFORMANCE THROUGH DATABASE CACHING AT APPLICATION TIER LEVEL

BOICEA, A[lexandru]; BADALAU, C[ezar] B[ogdan]; RADULESCU, F[lorin] & NICULA, A[drian] I[onut]

Abstract: *This paper analyses a technique for improving application performance through the implementation of database caching at the level of the application tier. The focus is Oracle In-memory database caching option for Oracle 11g and analyses the procedure of implementing Oracle TimesTen In-Memory Database as a database cache at the application tier for reduction of workload for Oracle database and shortening the application response time.*
Key words: *database, cache, application, tier, grid*

1. INTRODUCTION

The increasing usage of on-line services led to the growth of database sizes and also to users' demand for faster response time from applications. Therefore, the need for delivering critical and frequently accessed information within a shorter response time arises. A solution for this drawback in Oracle databases is the use of Oracle In-Memory Database Cache (IMDB Cache) at the level of application tier and provides three components to improve data access: TimesTen In-Memory Database, caching technology for frequently accessed tables cache and cached data consistency and a transactional data replication for cross – tier high availability.
The paper will present the procedure for data caching in Oracle databases by using Oracle In-memory Database Cache and test results of this solution in comparison with the classical 3 tier architecture.

2. MANAGING CACHE CONTENT

IMDB Cache is designed to cache table data from Oracle databases after which, exposes it to the application tier. IMDB Cache allows update on the cached tables data and synchronization between the native database data and cached data. This data is managed by the TimesTen In-Memory Database component and is associated with the Cache Agent which enables the data synchronization.
The Fig.1 outlines the context where IMDB Cache will be implemented. (Qiong, 2002)

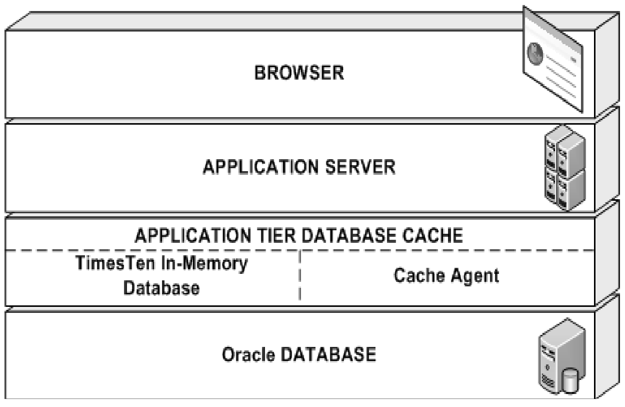

Fig. 1. Application tier database cache

A collection of database tables related through foreign keys that are transformed to IMDB Cache tables represents a cache group which may be defined through SQL Syntax.
For example, taking into account the diagram in Fig. 2, one may define a cache group composed of projects with a start date of 1^{st} *of January 2000* with an *eHealth* theme and a project duration larger then *36 months* and a second cache group made up by FP7 projects with a total project cost higher then *5 million euro* and *ongoing* status. The aforementioned cache groups may be cached on different nodes that run IMDB Cache enabling a distributed access to data that would result to a more efficient load balancing, therefore a faster response time.

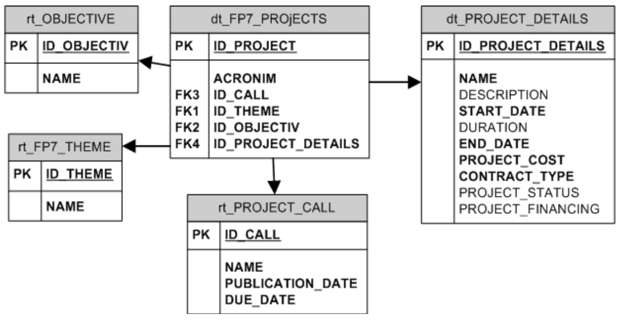

Fig. 2. FP7 projects database scheme

3. CACHE INSTANCE

A cache instance represents a set of uniquely identified and correlated records that is used to model complex objects. These instances provide the basis for cache loading and cache aging. Taking into account the database tables outlined in figure 2, all records in dt_PROJECT_DETAILS, dt_FP7_PROJECTS, rt_PROJECT_CALL, rt_OBJECTIVE, rt_GP7_THEME that belong to a specific project, are part of the same Cache Instance uniquely identified by a PROJECT ID called Cache Instance Key.
The data loading process may be carried out in two ways. The first one by explicit loading:
- loading an entire cache group at once,
LOAD CACHE GROUP fp7_projects COMMIT EVERY 256 ROWS
- loading cache instances through WHERE clauses (the WHERE clause selects a subset of the cache instance),
LOAD CACHE GROUP large_fp7_projects WHERE (dt_PROJECT_DETAILS.PROJECT_DETAILS >= 5000000) COMMIT EVERY 256 ROWS
- loading by ID (ID referring to cache instances to be brought in the cache).
LOAD CACHE GROUP large_fp7_projects WITH ID (8813, 221) (Cache Group Operations, 2010)
The second process is the dynamic loading that is appropriate to load cache groups that are to large to fit in the cache. This is associated with automatic Cache Aging implemented in IMDB

Cache considering the usage and the lifetime of the cache. Usage based aging follows the Least Recently Used algorithm that eliminates content which is least recently used in the cache group. Time based aging defines a time span for the content life cycle and requires that a table that is part of a cache group to have a timestamp column. The application is responsible for managing the timestamp column. Example of dynamic cache loading:

CREATE DYNAMIC ASYNCHRONOUS WRITETHROUGH CACHE GROUP new_fp7_projects (Cache Group Operations, 2010) *FROM dt_FP7_PROJECTS*
(ID_PROJECT NUMBER(6) NOT NULL,
NAME VARCHAR2(10),
ID_CALL VARCHAR2(50),
ID_THEME VARCHAR2(100),
ID_PROJECT_DETAILS NUMBER(6) NOT NULL,
PRIMARY KEY(ID_PROJECT))
(Cache Group Operations, 2010)

4. DATA CONSISTENCY MANAGEMENT

Cached data may be update in the IMDB Cache or in the Oracle database and the update is propagated automatically from the database to the cache and vice versa.

The tables in the cached tables groups should not be tables that are frequently updated.

The data updates should be limited to cache updates or database updates and afterwards the changes in the data should be transmitted correspondingly.

IMDB Cache applications provides the means to send SQL statements to the Oracle database or to the Cache group through a single connection to an IMDB Cache database by enabling PassThrough option that evaluates if either the SQL statement may be processed in the in-memory tables or in the Oracle database. This option is very helpful for specifying what types of SQL statements should be processed by the database and which should be processed in the IMDB Cache. For example, the updates should be passed to the Oracle database, while reads should be processed by the IMDB Cache. The update propagation from IMDB Cache to Oracle database is handled when a transaction updates a Cache instance in order to maintain data in the Oracle database synchronized with IMDB Cache. Therefore, after the transaction is committed, IMDB Cache propagates the updates to the Oracle database in the correct order. The update propagation from the Oracle database to IMDB Cache is done in three ways:

- the refresh operation is an explicit request from the application side to refresh all the content held by the Cache groups and it is the same as unloading the cache groups and followed by a load operation;
- full auto refresh specifies the time span between 2 automated refresh operations;
- incremental auto refresh updates only those records that have been changed in the Oracle database since the last refresh; this is useful for applications like customer support that may need to cache all incidents reported during a defined time span and it can be used in correlation with time based aging. When new incidents are inserted into an Oracle database, the incremental auto refresh will propagate to the in-memory cache tables. The incidents should have specified a timestamp column to be used in the evaluation of records to fit in the time window. Once the timestamp exceeds this period, the records from the in-memory cache tables are automatically aged out.

5. PERFORMANCE

In order to quantify the performance of IMDB Cache we have developed a JAVA application that interacts with the Oracle database through JDBC API. The JAVA application models the workflow of updating and retrieving records regarding FP7 projects and proposals. The technologies used for the test development were JAVA JDK 6, Netbeans 6.5.1, Oracle 11g, JDBC and IMDB Cache.

Changes of cached data are automatically propagated to the Oracle database.

The hardware configuration and all other configurations are identical for the tests carried out.

The difference between the execution of the stored procedures in the two scenarios was at the level of the functional architecture: the first onein the classical 3 tier architecture and the other one with the extra IMDB Cache component between data layer and bussiness logic.

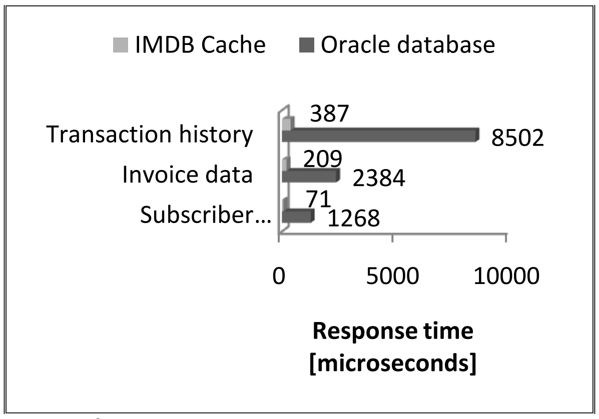

Fig. 3. Performance comparison

6. CONCLUSIONS

It is obvious from the results of the tests presented in Fig. 3, that In-Memory Database Cache brings tremendous performance gain for the application layer in web enabled applications. The decreaseof application response time is improved by at least one order of magnitude and for particular applications this represents a remarkable progress for users' real time experience. The preliminary results presented by this paper brings the opportunity to further research performance enhancement opportunities by correlating IMDB Cache with other Oracle database tuning techniques, but also to compare and test non-related Oracle caching solutions. Another point to continue this research activity is to further carry out tests in order to depict downsides of IMDB Cache configuration.

Also, the tests presented here will be taken to the next level by taking into account scenarios that involve large amounts of data, OLTP and OLAP databases, and a large set of cached tables. The objective will be to analyze results regarding application response time, data consistency and integrity and protection to system failures.

7. REFERENCES

Cache Group Operations. (2010). Retrieved May 24, 2010, from Oraclehttp://download.oracle.com/docs/cd/E11882_01 / timesten.112/e13073/operations.htm#CJAEFBHD

IMDB Cache. (2010). Retrieved May 12, 2010, from Oracle.com: http://www.oracle.com/technology/global/jp/ products/timesten/pdf/wp/wp_imdb_cache.pdf

IMDB Quickstart. (2010). Retrieved May 14, 2010, from Oracle.com: http://www.oracle.com/technology/products /timesten/quickstart/cc_qs_index.html

Introduction to IMDB Cache Concepts. (2010). Retrieved May 12, 2010, from Oracle.com: http://download.oracle.com/ docs/ cd/E11882_01/timesten.112/ e13073/ concepts.htm#BABEJFFC

Qiong, L. (2002). Middle-Tier Database Caching for e-Business. *ACM SIGMOD international conference on Management of data* (pp. 600 – 611). Madison, Wisconsin: ACM