International Journal of Computer Applications (0975 – 8887)
Volume 88 – No.13, February 2014

Object Caching Design Framework for Improving Data Access Performance in Enterprise Applications

Nilayam Kumar Kamila
Tech Architect
TATA Consultancy Services
G-Axis EPIP Area
Bangalore 560066 India

Renu Raghavan
IT Analyst
TATA Consultancy Services
28A High Street
Jersey City, NJ 07306 USA

Naveen Chalicheemala
Asst. Consultant
TATA Consultancy Services
G-Axis EPIP Area
Bangalore 560066 India

ABSTRACT
In today's technological world, applications are designed to communicate across different interfaces and databases with high speed mode. The data communication between applications and the database needs accuracy as well as quick fire. There have been many designs implemented to reduce the transfer time. Caching is one of such attempt to reduce data access time. But it becomes difficult to cache database results (dynamic data) when the database is manipulated by various external applications. The caching technique provides a faster way to access data, but at same time it should capable to provide the data accuracy. There is no such generic model or design available till today which could capable to cache the highly dynamic data [4]. Here a model is presented, where by implementing this design, the data results will be cached with complete data accuracy in application level and will reduce the repetitive transmission time and hence increase the overall performance of the system.

General Terms
Application Caching Design

Keywords
Database Caching, Caching framework, Data Caching, Application Performance, Performance Improvements

1. INTRODUCTION
The Data Caching [1] is an existing technique to cache the application objects. Many concepts and frameworks have been evolved to cache the data and use repetitively in the application instead of invoking remote calls, file read, service calls etc. [2]. But for dynamic data(which changed most often by different applications or users), applications more rely on calling the remote calls than using the cached data as the cached data and the source data will not be in synch, if the data is dynamic in nature.

Most applications use cache technique to make static data available throughout the application scope. Also in most cases, developers or designers use their own style of caching rather than to implement any existing framework [2]. This paper explains different scenarios of the caching mechanism approach in order to resolve the caching issues for all types e.g. static, dynamic data. The proposed approach is based on a basic pluggable technique with few changes in database level (study applied and verified only to Oracle and MS SQL; other database change is under further study) which will provide a better performance over the repetitive remote calls.

This solution provided in this paper is well suited to the enterprise level console, windows and online based applications. We have also distinctively taken multiple data structures into consideration to improve the caching performance.

In next section we discussed existing problem and section 3 focuses on the proposed solution approach. The caching frame design is explained in Section 4. The performance analysis on different data counts is presented in Section 5. The inference and future scope of this paper is discussed in section 6.

2. THE EXISTING PROBLEMS
Enterprise applications retrieve data from database for which the application hit database multiple times. The primary problem in this respect is that, the application has high response times which need to be reduced. For this purpose, caching is adopted to hold frequently retrieved data in application objects, but the problem still exists as there is no model is framed to handle different nature of data e.g. static and dynamic data.

All business or enterprise applications in today's world are communicating with database. The data is retrieved from database to be displayed and modified by the users through online or windows based screens. In this process, the application is retrieving same data contents multiple times from database even if there is no change or minimal change on database. The enterprise applications are referring application and business related data multiple times by multiple users, there is a more number of database call which results a delayed response time. Another problem that arises is, if all data is cached in application without considering the data nature, then application data and database data will be out of synch frequently.

In order to abstract the complete application problem on caching point of view, the data in database is categorized into three categories.

i. Static data: In this category, data which are never changed during the complete life cycle of the application. These static data are never modified by application or by any other external application or users e.g. lookup tables data, constant data etc.

ii. Self controlled data: The data which are modified only by the application. These types of data are never modified by the external applications e.g. application self transaction data.

iii. Fully Dynamic data: Fully dynamic data are dynamic in nature and could be modified by external applications and or by db users.

All the above category data in database are playing their respective roles in applications. Application data and database data at any specific instance of time must be in synch to make the application reliable and consistent; otherwise the application will show up stale data which in turn gives incorrect results and inconsistent database and application state.

In next section the solution approach will be visualized to make all above categories data to be in consistent state through the object caching methodology.

3. PROPOSED SOULTION APPROACH

The object caching technique is implemented to resolve all types of above discussed problems. The proposed approach provides necessary setups for caching and processes to identify whether the data available in cache is db synch up data or stale data. Let's discuss the proposed caching design solution for different category of data.

3.1 Object Caching – Static Data

The static data caching methodology is a simple technique which most of the designers or developers implement in their web or windows application. But the developers' technique is formalized in this proposed approach. All static data related database stored procedure name key will be put in the CacheConfig.xml. The application now loads the CacheConfig.xml files and will compare for each application's database request. If the request matches with an entry in DataConfig.xml, it will then attempt to search the results in Object Cache area. In first time data search, it will not find the result set in the caching area, so it will retrieve from database and will put in the cache area, and then return the result set to the calling unit. In next successive hit, it will search the cache area and will find the result set and directly return the result set without invoking the database call.

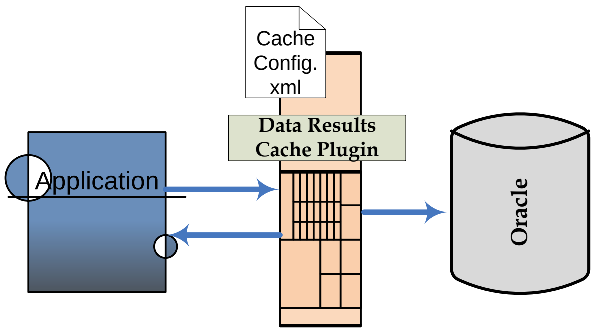


Figure 1. Design Solution for Object Caching – Static Data

As shown in figure – 1, application will always try to find the desired result set object from the data results cache area. If application does not find the object in cache area, then it will hit the database; and after successful retrieval, it will store the result set object in cache area. This way the application becomes intelligent as the application life progress.

The approach also includes loading all the data results for static data at the time of application initialization. This way when request comes in runtime, then that result set could be found from cache immediately and is returned back. This strategy could be adapted by setting the flag in ConfigCache.xml.

3.2 Object Caching – Self Controlled Data

In this section it will be seen that how to cache the result set of the database for which the database is modified by the application only. For this purpose a CacheLive (refer figure 2) area is defined in the proposed design solution. The CacheLive area remains in memory and mainly contains the information about the Data Objects with their dirty flag information. If any time the application is invoking to update or delete or insert data in the database, then the corresponding dirty flag in CacheLive area is made to true(means the CacheLive dataset and database dataset are not in synch). When next invocation for the data occurs, it will verify the CacheLive information, and if found dirty (dirtyflag=true) then it will retrieve the data from database and store in the CacheLive area and return the fresh results to the calling unit.

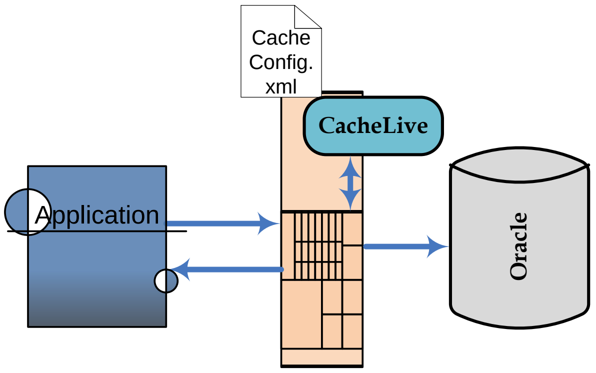


Figure 2. Design Solution for Object Caching – Self Controlled Data

CacheLive area is used to maintain the dirty flags as shown in table 1, which provides the information whether the application hold cached data are in synch with database or not. If for any cached information, the data is out of synch, then CacheLive information makes the dirtyflag to true.

Table 1: Data Structure for CacheLive Area

CacheLive	
DataObjectName	**dirtyflag**
GET_FUND	true
GET_CASH	false

At initialization time all the entries for dirty flag in CachedLive made to true. So in first database result set search the framework will retrieve the data from database and make the corresponding entry as false. The next time onwards, if there is a result set retrieval, then the framework will retrieve the result set either from cache area or from database depending on the CacheLive dirty flag information.

3.3 Object Caching – Fully Dynamic Data

This type of data is being modified by the external applications and (or) may be manually through the database insert, update or delete statements. As the application has no control over this category of data, so a separate caching mechanism is adopted to handle such dynamic contents. There are basically three steps to establish this caching mechanism.

3.3.1 Data Update Trigger

There need a trigger mechanism implemented on the tables whenever there is an insert, update or delete. So when there is a modification to the data tables, this data manipulation triggers will be invoked. In oracle, this trigger will put a message e.g. database named object's information to a queue known as Oracle Advanced Queue(refer figure 3)[6]. In MS

SQL, this trigger will invoke a SQL Job through SQL Job Agent[5].

3.3.2 Oracle Event Job Scheduler

There is a package in Oracle known as DBMS_SCHEDULER which will poll the message whenever there is a message available in Oracle Advanced Queue. This scheduler then invokes an application batch job.

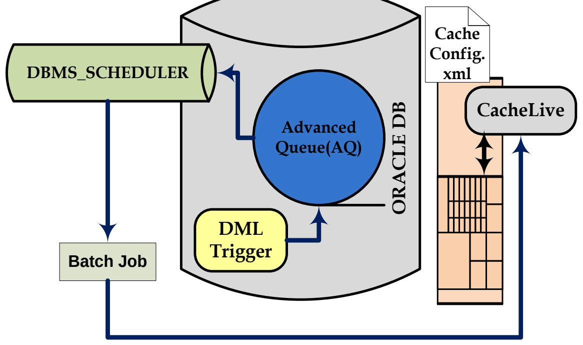


Figure 3. Design Solution for Object Caching – Fully Dynamic Data in Oracle database

3.3.3 Batch Job CacheLive Communication

The batch job would run either on database local system or on the remote system where the application is running. This batch job will update and refresh the CacheLive area for the respective data object. This batch job will run in thread mechanism to handle concurrent requests from the DBMS scheduler. However the update to CacheLive will be handled programmatically through the synchronized block with respect to data object.

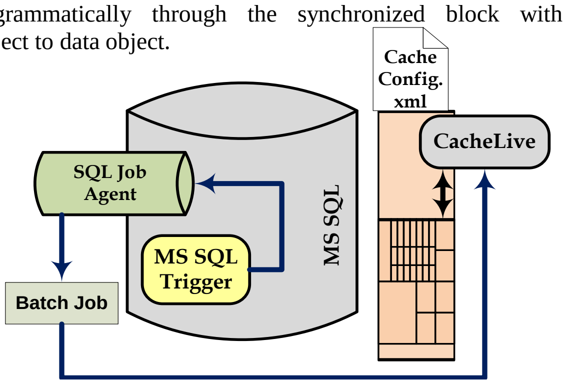


Figure 4. Design Solution for Object Caching – Fully Dynamic Data in MS SQL database

Figure 4 depicts the approach in MS SQL Server. When there is a modification on data set in MS SQL database, it will invoke the trigger and trigger will invoke the named job in SQL Job Agent. The SQL Job agent has the capability to invoke the external batch job which will refresh the CacheLive in the proposed design.

4. CACHING FRAME DESIGN

In this section the low level caching frame design will be overviewed and how best it could be pluggable to new or existing applications. There are several algorithms out of which some are presented here to explain the basic structure of this solution approach. As in previous section, It is seen that there is an xml file CacheConfig.xml for configuration. The detail is shown next. Here it is shown for static data, and similar fields could be used for self controlled and fully dynamic data. When the application will be initialized, then this CacheConfig.xml will be loaded to a static collection object. The application is then refer to that static collection object each time to take a decision whether to lookup the results from cache area or from database server.

```
<dbload-on-statrtup>-1</ dbload-on-statrtup>
<Staticdata>
 <DBobjects>
  <StoredProcs>
   <Name>SP_GET_STATDATE</Name>
   <Name>SP_GET_ENDDATE</Name>
  </StoredProcs>
  <Functions>
   <Name>FN_CAL_PERIOD</Name>
  </Functions>
  <Tables>
   <Name>TBL_COMPANY</Name>
   <Name>TBL_INFRASTRUCTURES</Name>
  </Tables>
 </DBobjects>
</Staticdata>
```

In algorithm 1, It is shown that the CacheCofig.xml file is completely loaded into the application. There as three distinct sections e.g. static, self controlled and fully dynamic in the xml file and each section again contains the stored procedure, function and table objects names.

Algorithm 1. initCache()

```
load CacheConfig.xml to CacheConfig Object.
if(dbload-on-statrtup)
  for(each staticDBObjects in CacheConfig)
    loadDBDatatoAppOjects(cachObjectName);
  endloop
  for(each selfConDBObjects in CacheConfig)
    initializeCacheLive(selfConDBObjects);
    loadDBDatatoAppOjects(selfConDBObjects);
  endloop
  for(each dynamicDBObjects in CacheConfig)
    initializeCacheLive(dynamicDBObjects);
    loadDBDatatoAppOjects(dynamicDBObjects);
  endloop
end if
```

There is a tag named as “dbload-on-startup” which flags whether to load the static and CachLive data to the application at the start up time. If it is configured to set as -1 then it will not load at startup time. It will load the db objects into cache during the respective runtime db requests.

Let’s discuss how this design will work during the run time db requests. The db request item will be compared with CacheConfig collection. If the db requested item is found in CacheConfig, then it will check if the type is static or self controlled or fully dynamic.

If the db request is a static type then it will attempt to retrieve the data from static cache area through the lookupCache() method. In lookupCache(), the application will verify whether the data set is available in Cache area, if not then it will retrieve the data from database and will update the static cache.

Algorithm 2. retriveDataObjects(dbRequest)

```
if(CacheConfig contains dbRequest)
  if(typeOf(dbRequest) is static)
  | lookupCache (dbRequest);
  else  if (CachLive contains dbRequest)
        checkDirtyness(dbRequest);
        if(dirty)
           invokeDBOperation(dbRequest);
           updateLiveCache();
        else
           lookupLiveCache(dbRequest);
        end if
  end if
else
   invokeDBOperation(dbRequest);
   //Item not eligible for Caching
end if
```

Similarly if the db request data is for self controlled type or fully dynamic type, it will verify the CacheLive area and will verify the dirtiness of the requested object. If it is in synch with database (i.e. dirtiness is false) then it will retrieve from Cache, otherwise it will invoke the database call and will update the CacheLive accordingly.

5. PERFORMANCE STUDY

This section shows how the proposed design is performing against the direct database call. It is also studied and verified that the other existing caching frameworks e.g. EHCache[7], JBoss Cache[8] are compatible with the proposed design. Here it is enlisted few results by hitting the database multiple times for static data which gives excellent results for proposed design as shown in Table 2.

Table 2: Response time against database hits

Number of db hits	With Cache (in ms)	Without Cache (in ms)
1	13456	12659
2	7865	11985
4	2774	12106
8	1398	11426
12	934	11201
15	758	11339
20	558	11622
30	371	11394
40	284	11431
50	235	11640

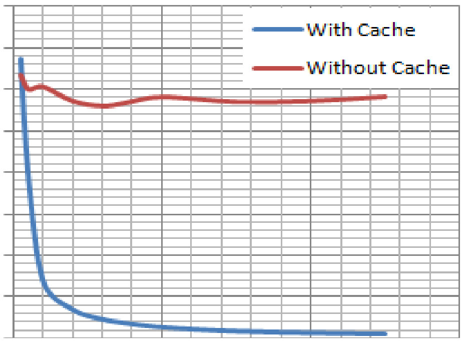
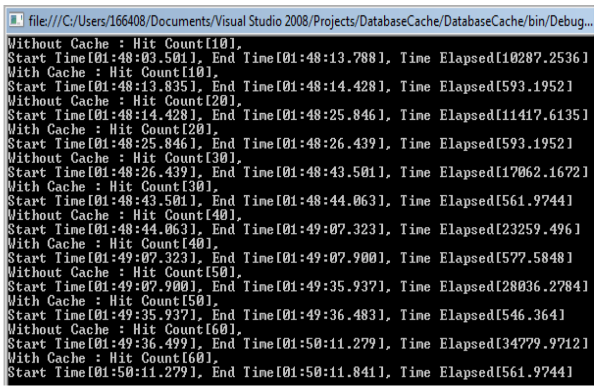


Figure 5. Performance graph number of database hit vs time

Figure 5 shows the performance of cached data against the non-cached data. The initial performance is slightly less in case of cache design as it load the CacheConfig.xml and compare the db request with the cache area. The first request always hit the database and store the result set in cache area. However as the life of application increases, the cache system performs better than the non-cached design.

The proposed design model is simulated in .NET application and the elapsed time is captured with proposed cache model and without cache model for different types of data. The captured data are shown in below console figure (ref figure 6).



```
file:///C:/Users/166408/Documents/Visual Studio 2008/Projects/DatabaseCache/DatabaseCache/bin/Debug...
Without Cache : Hit Count[10],
Start Time[01:48:03.501], End Time[01:48:13.788], Time Elapsed[10287.2536]
With Cache : Hit Count[10],
Start Time[01:48:13.835], End Time[01:48:14.428], Time Elapsed[593.1952]
Without Cache : Hit Count[20],
Start Time[01:48:14.428], End Time[01:48:25.846], Time Elapsed[11417.6135]
With Cache : Hit Count[20],
Start Time[01:48:25.846], End Time[01:48:26.439], Time Elapsed[593.1952]
Without Cache : Hit Count[30],
Start Time[01:48:26.439], End Time[01:48:43.501], Time Elapsed[17062.1672]
With Cache : Hit Count[30],
Start Time[01:48:43.501], End Time[01:48:44.063], Time Elapsed[561.9744]
Without Cache : Hit Count[40],
Start Time[01:48:44.063], End Time[01:49:07.323], Time Elapsed[23259.496]
With Cache : Hit Count[40],
Start Time[01:49:07.323], End Time[01:49:07.900], Time Elapsed[577.5848]
Without Cache : Hit Count[50],
Start Time[01:49:07.900], End Time[01:49:35.937], Time Elapsed[28036.2784]
With Cache : Hit Count[50],
Start Time[01:49:35.937], End Time[01:49:36.483], Time Elapsed[546.364]
Without Cache : Hit Count[60],
Start Time[01:49:36.499], End Time[01:50:11.279], Time Elapsed[34779.9712]
With Cache : Hit Count[60],
Start Time[01:50:11.279], End Time[01:50:11.841], Time Elapsed[561.9744]
```

Figure 6. Proposed Cache Model Simulation Run - Proposed Cache Model elapsed response time Vs non-cache elapsed response time in .NET Application.

The resulted output is put on the graph which shows (refer figure 7) that the proposed cache model for db results is performing better than without cached model. Here in this simulation the total response time elapsed is taken during the data retrieval from proposed cache model vs non-cached system. It is observed that the elapsed response time is increasing in both the cases, but the response time gap is more and more when the application is hitting the databases more number of times.

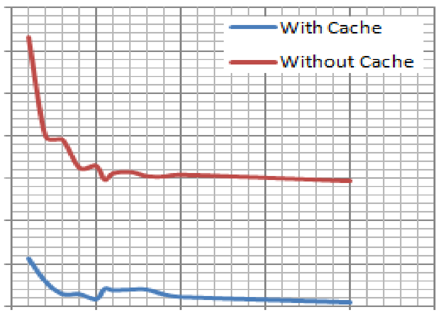
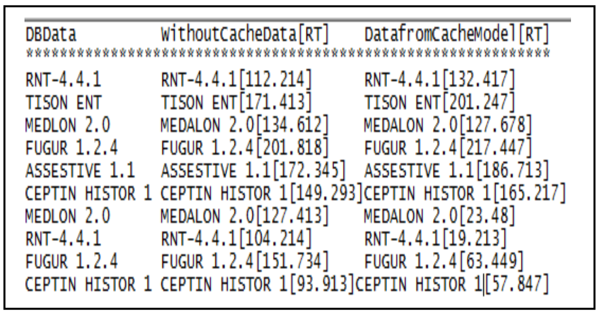


Figure 7. Performance graph number of database hit vs time in .NET Simulation (Static Data)

The accuracy of the data is also verified and the data retrieved from both the model is accurate with the Database data. In reference to accuracy it is also observed that(ref figure 8) the data when retrieved for the first time from data base is almost same time or slightly more time because of the model to gather the cache information and to verify whether the data available in cache area or not. But subsequently, the response time is reduced as compared to the non-cached system as the application progresses.

It is also compared the proposed caching performance for self controlled data types. Figure 9 shows how the proposed model performs against the self controlled data.



```
DBData          WithoutCacheData[RT]     DatafromCacheModel[RT]
*****************************************************************
RNT-4.4.1       RNT-4.4.1[112.214]       RNT-4.4.1[132.417]
TISON ENT       TISON ENT[171.413]       TISON ENT[201.247]
MEDLON 2.0      MEDALON 2.0[134.612]     MEDALON 2.0[127.678]
FUGUR 1.2.4     FUGUR 1.2.4[201.818]     FUGUR 1.2.4[217.447]
ASSESTIVE 1.1   ASSESTIVE 1.1[172.345]   ASSESTIVE 1.1[186.713]
CEPTIN HISTOR 1 CEPTIN HISTOR 1[149.293]CEPTIN HISTOR 1[165.217]
MEDLON 2.0      MEDALON 2.0[127.413]     MEDALON 2.0[23.48]
RNT-4.4.1       RNT-4.4.1[104.214]       RNT-4.4.1[19.213]
FUGUR 1.2.4     FUGUR 1.2.4[151.734]     FUGUR 1.2.4[63.449]
CEPTIN HISTOR 1 CEPTIN HISTOR 1[93.913]CEPTIN HISTOR 1[57.847]
```

Figure 8. Data accuracy verification against database data with response time details

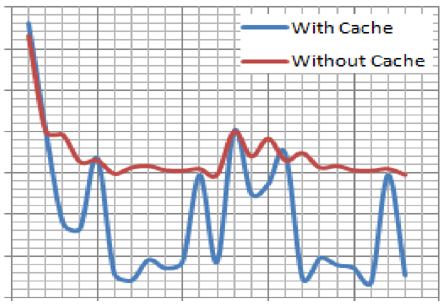


Figure 9. Performance for Self Controlled Data in proposed cache vs non-cache system

In figure 9, it could be seen that there is some instance where cached system response time equals with non-cached response time. This is due to the cached system hit the database where it did not find the data in cache live area or it might find the dirty flag as true for the requested data. However, on overall, the proposed cached system performing better than the non-cached system.

6. CONCLUSIONS

This proposed design is providing a better performance over the existing design of repetitive database call. The static, self controlled and fully dynamic data for oracle and MS SQL based applications is considered in this paper which performs with the proposed approach's expectation. In addition to that, different existing caching framework is included and simulated to measure the compatibility which shows clear distinctions that the proposed design is compatible with existing caching and also performs in a better scale.

It is still in research process to include other (DB2, Informix, MySQL etc) databases and to include the native joins by taking two datasets into considerations which will enhance the application performance in case of dual of multi - db calls application requirements. The cached result set join also include the dataset from different databases e.g. joining oracle based data set with MS SQL based dataset; as part of the future scope of this design approach.

7. ACKNOWLEDGMENTS

We are very much thankful to TATA Consultancy Services Ltd, and the managers to provide us the infrastructure, motivations and continuous guidance along with the technology base which really help us in all way to write this paper. We are thankful to Neelakanteshwar Palla, Hari Chennuru, Suneel Satrasala, Vishnu Varadhan Kalyanam, Manickam Thriponithura and other members for providing the support all times to better execution of this proposed framework. Never the less, the continued support from our family is recommendable, without which this could not take a complete shape in this design implementation.

8. REFERENCES

[1] H. Opfner, S. Wendland and E. Mansour, "Data Caching On Mobile Devices The Experimental MyMIDP Caching Framework". http://elab.ws/portal/files/publications/icsoft09-cach.pdf.

[2] Nilayam K Kamila and Prashanta K Patra, "An Extensive Data Caching Framework Approach for Enterprise Application. In Journal of Theoretical and Applied Information Technology. http://www.jatit.org/volumes/Vol52No1/11Vol52No1.pdf

[3] Examples of Using the Scheduler. http://docs.oracle.com/cd/B28359_01/server.111/b28310/schedadmin006.htm

[4] Open Source Cache Solutions in Java. http://java-source.net/open-source/cache-solutions

[5] EHCACHE. http://ehcache.org/documentation/code-samples

[6] JBoss Cache Users' Guide. http://docs.jboss.org/jbosscache/3.2.1.GA/userguide_en/html_single/