Journal of Theoretical and Applied Information Technology
10th June 2013. Vol. 52 No.1
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN EXTENSIVE DATA CACHING FRAMEWORK APPROACH FOR ENTERPRISE APPLICATION

[1]NILAYAM KUMAR KAMILA, [2]PRASHANTA KUMAR PATRA
[1]Asst. Consultant, TATA Consultancy Services, Florida, USA
[2]Prof. & Head, Department of CSE, College of Engg. & Tech, Bhubaneswar, INDIA
E-mail: [1]nilayam.kamila@tcs.com, [2]hodcomputer@yahoo.co.in

ABSTRACT

There are a number of approaches, which have been developed and are still in process of development, to regulate and frame the data transfer to/from the servers or storage base. The developers are still facing number of difficulties in terms of time required in fetching or storing the data in remotely located servers during the development (coding and unit test) phase. The remotely located server could not be replicated or placed in local area network due to high security concerns and unavailability of proper infrastructures. We are presenting a new framework approach which can be utilized to cache the remote data and hence the development process could be made faster than the usual traditional development process.

Keywords: *Caching, Data Caching, Caching Framework, Development through Caching*

1. INTRODUCTION

The Data Caching is a key technique to reduce the response time so as to improve the overall performance of the software application [1]. This technique is widely used in many contexts e.g. web application, database technology, proxy server etc. With this technique, application users can interact with the system and in turn the CSI (Customer Satisfaction Index) could be increased at its desired level [6].

There are many e-commerce or enterprise level application sites which are mostly based on data-driven design, but some of these data-driven websites can't be developed quickly because of the development location constraints. The servers are located in different geographic regions and the developers are located in different geographic regions. In this case, the application developers or designer could implement own and private data-caching mechanism to ensure that all the application related data objects are retrieved appropriately to the application[4]. A better way to retrieve the application data is to use caching objects, which cache the data when it is retrieved from the server. The caching framework discussed here is a more generalized way to resolve the data fetching response time issue in development phase(coding and unit testing), in reality.

There are also a lot of difficulties faced by the application developers, where they have to develop the application in a restricted availability of infrastructure (because of the data security concerns, cost factors etc.), and hence they need to depend on the remote systems to get the data. The application is initially developed in an environment where the infrastructure was available, and there is hardly any data fetching response time issue. Later on, when the same application is handed over to a set of developers who are in a different geographic region (due to minimize the cost of development and maintenance), the data fetching issue comes in large extent to the picture. Our proposed approach is an attempt to resolve the issue of the application developers for the multi-tier architecture based applications. The solution presented in this paper refers to a two tier and three tier architecture based applications. This approach could be extended to n-tier architecture based application and the performance will be enhanced in similar alignment. In this paper we have provided the approach for the Data Caching framework for online web application. The same line of approach could be designed and implemented in the console or windows based application. The discussion of different types of implemented data structure model on this frame work, presented in this paper, provides a distinctive improvement over the initial designs.

In Section 2, the related works are discussed. Section 3 focuses about the existing development process. The proposed approach is explained with algorithms, design and data structure models etc. in Section 4. The data structure model, system design and the data flow is shown in section 5, 6 and 7 respectively. The mathematical analysis and simulation analysis on performance is presented in section 8 and 9 respectively. The inference and future scope of this paper is discussed in section 10.

2. THE RELATED WORKS

In recent advancement of the technology, there are many cache frameworks [7] are evolved in various scope. Some of the most common and advanced frameworks are discussed in this section. The 'SwarmCache' is a distributed cache which caches the references in a distributed network. There is another Object Cache based framework, known as 'ShiftOne', which built on a java library with several strict object caching policies. It also confirms to light framework based cache system. Java Caching System 'JCS' is another distributed caching system written in java for server-side applications. This framework is well suited to the distributive system. 'OSCache' is a caching framework which has servlet-specific features and persistent in nature which leads to the data security concerns. The other type of cache 'Whirlycache' is a fast, configurable in-memory object cache for Java. This is primarily designed to cache the object by querying or calling stored procedure to database, but the other type of server calls e.g web service calls, rmi calls etc. are out of scope.

In most of the caching approach, the attempt is made to implement the frame work in production environment. The data security concern is least taken care of. These cache systems are mostly incompatible with the system where the production environment needs the real time data at every target server request.

3. EXISTING DEVELOPMENT PROCESS

We will focus on the ‘Development’ phase of the Software Development Life Cycle (SDLC) where ‘Implementation’ is the part of the process in which software developers actually write codes for the real time application. Unit testing is also playing an important sub-phase in the software development process; though this phase comes under the ‘Development’ phase. In next phase i.e. testing phase, defects are caught and developers attempt to fix the errors as soon as possible in order to avoid the delay in application delivery.

In existing tradition style of development, the application developer develops the application with the local environment setup against the remotely located interface servers e.g. System Integration Test (SIT), or Development (DEV) Servers. So once the whole or part of a requirement is completed, the developers will test the code in local setup by hitting to the remotely located DEV or SIT Servers. Each error found in this process, the same development process will be repeated, until developer certify the code is ready for the SIT environment. During unit test, each time hitting to the remote server will cost more in terms of the response time and network resources.

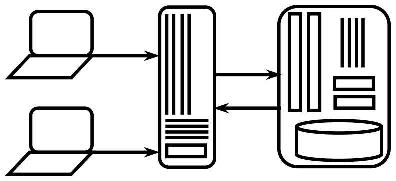


Figure 1. A sample two - tier application model

Figure 1 depicts about a sample two tier application model where the client application (web browser/windows/console based application) accessing the web/application server and the web or application server in turns reach to another target server to fetch the required data for the intended request. In this model the target server may be another application server or may be a database server. Let’s now discuss about another sample model in three-tier or n-tier architecture to visualize the data flow in the advanced application.

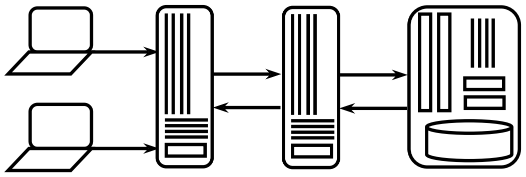


Figure 2. A sample 3-tier application model

In figure 2, we have complex data flow model where the data requested by the user to the web or application server will again to be requested to another middle ware server, which again reach to database or application server to get the exact data. In this model there is another layer (technically known as 'tier'), which is involved in the data flow model of the application. So the more number of layer present in the application architecture, it will be more complex, and the developers need to invest more time to unit test the modification in existing application or to develop a new application.

4. PROPOSED CACHING FRAMEWORK

4.1 Overview

The data caching framework approach is to provide the necessary setup so that the application will be able to fetch the data from the cache object

Journal of Theoretical and Applied Information Technology
10th June 2013. Vol. 52 No.1
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

in each time hit to remote servers. The first time hit would fetch the data from the real time remote DEV/SIT servers and then it will store in cache, and in next time onwards, when user fetch the same request data, it would return the cached data instead of hitting the expensive target. This framework is a clean approach to be configured in any tiered based application architecture and would save the time in maximum, and thus improves the performance in application development phase.

4.2 Working Model

The working model is designed in such a way that, this will work in both development and production environment. As per the assumption, this framework must not cache any data in production environment. There is a configuration file maintained, which will redirect the request to the real-time servers at its first hit and then it will redirect to the **D**ata **C**ache **I**nterface (DCI) from second time onwards. Let us discuss about the framework in detail and then we will fit this approach in a typical two and three tier application model.

There are three steps for this proposed framework model. They are as follows.

i. ConfigCache.xml Configuration
ii. Data Cache Interface - DCI layer
iii. Application layer Modification

4.2.1 ConfigCache.xml Configuration

The ConfigCache.xml is an xml file which controls the framework at its execution time. This file contents are as follows.

1. ENVIRONMENT – DEV/SIT/STAGE/PROD
2. CACHE_REQUIRED-WITH_CACHE or NO_CACHE
3. APPLICATION_URLS
4. Any Caching URL is available to the application
    a. CACHING_URL_LOAD_REQUIRED – YES/NO
    b. CACHING_URL_LISTS
5. DATA_FAILURE_LOAD_RETRY – YES/NO DATA_FAILURE_LOAD_URL –The Remote alternate Urls, if any

4.2.2 Data Cache Interface - DCI layer

This Interface plays a vital role in the Data Caching framework. This is basically a set of class files packaged in a module and to be referred by the application layer. The following are three basic elementary tasks implemented in this DCI layer.

a. loadConfigCache()
b. processReroute()
c. recoverFromFailCacheURL()

Let’s focus each task’s brief functionality with their algorithmic approach.

The loadConfigCache() task of the DCI layer is used to load the framework’s control file i.e. ConfigCache.xml and set the additional application configuration parameter for the NON-PRODUCTION environment. If the environment defined in the ConfigCache.xml file as PRODUCTION, or the file is not available, or any of the exception scenario happens, it will unset the additional configuration parameters.

Algorithm 1. loadConfigCache()

```
Step 1.  retrieveAppInitParameters()
Step 2.  If (isLoadCacheRequired)
Step 3.  | Try
Step 4.  |   | loadConfigCacheXML();
Step 5.  |   | If (ENVIRONMENT != PROD)
Step 6.  |   |   | setAdditionalAppConfig();
Step 7.  |   | Else
Step 8.  |   |   | unsetAdditionalAppConfig();
Step 9.  |   | End if.
Step 10. |  Catch(Exception)
Step 11. |   |     unsetAdditionalAppConfig();
Step 12. | End Try.
Step 13. Else
Step 14. |     unsetAdditionalAppConfig();
Step 15. End if.
```

The unsetAdditionalAppConfig() is to release the expensive resources and (or) frees unused memory object to improve the system performance in a better way.

The processReroute() is another task embedded in the DCI layer to take the intelligent decision such as, when to return the data from cache and when from remote server. In this proposed model, the first time data load is made from the remote server and successively from the next hit; it will load the cached data.

As this approach is meant for the development phase, so the cached data will fulfill the developer’s data requirement in coding and unit testing phase.

Each type of remote server request is associated with an object key, which is used to identify the first time request vs the old request. This key is stored in a key-value based library object as key, and the real object is stored in the respective object area(Refer section 5). The first time request will be associated to a new key and will be retrieved from the remote server, and the successive old request will be tried to get the object from the object cache.

Algorithm 2. processReroute()

```
Step 1.  If(isObjKeyFound())
Step 2.  |     loadFromCache()
Step 3.  Else
Step 4.  |   If(isURLCaching())
Step 5.  |   |   Try
Step 6.  |   |   |   loadFromCacheURL();
Step 7.  |   |   |   setObjectKey();
Step 8.  |   |   Catch(Exception)
Step 9.  |   |   |   If(isFailureCacheAvailable())
Step 10. |   |   |   |   recoverFailCacheURL();
Step 11. |   |   |   |   setObjectKey();
Step 12. |   |   |   Else
Step 13. |   |   |   |   loadFromAppURL();
Step 14. |   |   |   |   setObjectKey();
Step 15. |   |   |   End if
Step 16. |   |   End Try
Step 17. |   Else
Step 18. |   |       loadFromAppURL();
Step 19. |   |       setObjectKey();
Step 20. |   End if
Step 21. End if
```

If any failure occurs while retrieving the object from the cache, then it will try to recover the same from a second level cache, known as failure recovery cache interface system. If it still fails, the interface will retrieve the data from the remote SIT/DEV server.

Failure Recovery load is an additional mechanism provided to this framework to establish a robust data caching system. The failure recovery URLs made available in the ConfigCache.xml to make this approach to work in the appropriate forum. The failure recovery URLs would be the alternate URLs provided to the application which are intended to make the system to work even the primary level of cache system fails.

The framework provides the cache object at runtime where the data are not cached in any persistent memory, file system or database. This excludes the point of the security concerns on data during the development phase.

4.2.1 Application Layer Modification

In this setup, we need to incorporate the caching frame work in the application code. This will be accomplished in the remote line calls where the developers feel that the respective piece of remote call consumes maximum time to retrieve the remote data. This extra piece of code in application will fetch the real data or cached data depending on the proposed framework configuration.

The same modified piece of code for the caching framework could be incorporated according to the flag set in the application configuration file i.e web.xml in JAVA web application/services and web.config in ASP .NET web application/services. Let's see, how the application code is to be modified to establish this framework through the following algorithm.

Algorithm 3.processApplicationRequest()

```
Step 1.  retriveInitalizedParameter()
Step 2.  If(PROD)
Step 3.  |  loadFromAppURL()
Step 4.  Else
Step 5.  |  loadConfigCache()
Step 6.  |  If(WITH_CACHE)
Step 7.  |  |     processReroute()
Step 8.  |  Else
Step 9.  |  |     loadFromAppURL()
Step 10. |  End if
Step 11. End if
```

If the web.xml or web.config file init parameters e.g. *isProd, isLoadCacheRequired,* etc. are set to true or false in order to enable or disable the data caching framework impacts. These parameters are to be set appropriately during the application deployment in the DEV, SIT and PRODUCTION environment. However, even it is not set in the configuration file, then a secondary level of verification is done (refer algorithm - 1), so as not to load cache data in PRODUCTION environment. This could be extended to ensure for other lower environments according to the case to case in application development phase.

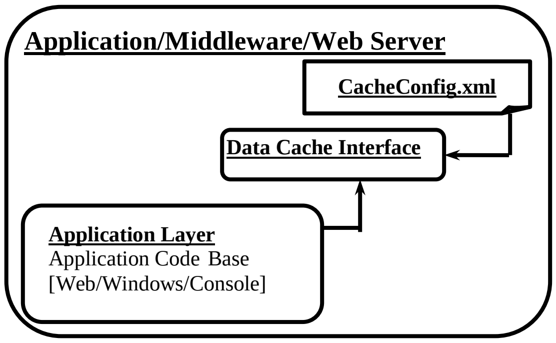
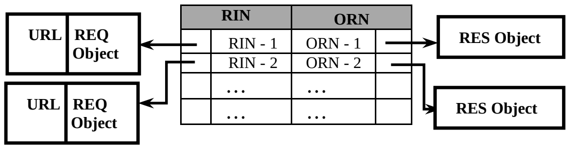


Figure 3. Working Model of Proposed Data Caching

Figure 3 shows the complete working model of the different layers' view in its scope. The application layer will work according to its modification to adheres the framework in a tightly coupled way and work with respect to the control file (ConfigCache.xml) key set, which is intelligently routed and cached in the DCI layer.

5. DATA STRUCTURE MODEL

This framework was initially implemented with a list based data structure model, where each object in list contains the url, request object and response object. This worked without any performance issues when the number of requests were less. In list model, it was found that, though the framework has a better performance over the non-cached system, but still the searching process takes more time to get the required response object from the cache area. So the implementation was promoted to a key-value pair Hash Map based data structure, where the key was set as Request Identification Number (RIN) and the value was set as Object Response Number (ORN) as shown in figure 4.



Fiaure 4. Data Structure Model

The RIN values were sorted and indexed to search the request. This reduces the searching time and improves the system performance for comparatively more number of requests. Later, it is realized that the performance could be made more efficient for large number of requests through the implementation of Tree Map where the search process will be accomplished through the B-tree search technique. The Tree Map implementation of this framework reduces the search time to O(log(n)) from O(n). The further implementation of the more complex and efficient data structure model e.g. B+ tree, B* tree, AVL tree search technique, to resolve the searching issues are still under progress.

The other factor which is affecting this framework is the amount of data to be set in the request object and the response object. In order to reduce the time to set the bulk data in run time, the pattern recognition concept is taken into the picture. The request and response patterns are now under the research study for various types of request e.g. http, soap, xml, iiop etc. The improvement of the data structure model to resolve the bulk data copy in request and response object will lead to the implementation of the pattern recognition and fuzzy logic concept, which is the future version of this framework.

6. SYSTEM DESIGN

In this section we will discuss about how and where we will fit each layer of this framework in the application. The DCI layer, we can create dynamic libraries (.dll) in case of .NET based application or an archived resource (.jar) in case of a JAVA based application. This DCI layer module could be put in any common location where all other deployed module could take the advantage of this caching framework or may be located in the private folder of the application in target. Below figure shows how this framework is established in two-tier architecture based application.

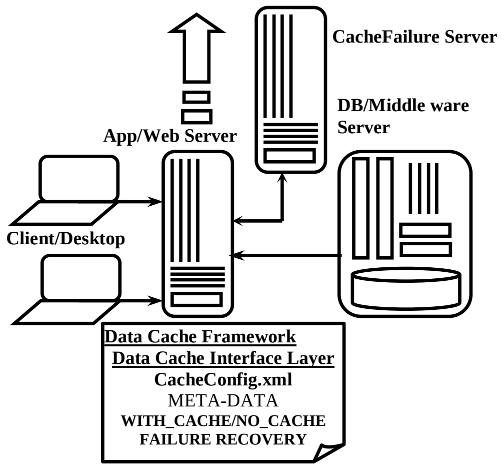


Figure 5. Data Caching Frame on 2-tier application model

In figure 5, it is shown that, the framework is packaged with the class files of Data Cache Interface - DCI layer and the control file ConfigCache.xml.

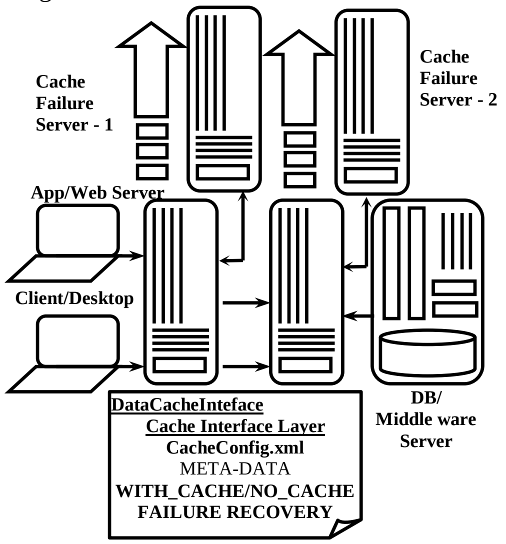


Figure 6. Data Caching Frame on 3-tier application model

The dynamic library or the archive resource file will contain only the native or class level codes, and the ConfigCache.xml resides outside the package, where the developer change the contents according to the requirement.
The design of cache failure system is an approach defined in this framework; which could be configured to ensure that the system will work even the primary level caching fails. However the failure server url(s) could be server application(s) that the developer could maintain in Web/Middle ware server or in same system. Hence there is no additional infrastructure required to implement a second level cache in this design approach. Figure 6 is an extension of 2-tier application architecture to show the design of the framework in a 3-tier application.

7. APPLICATION DATA FLOW

This section will briefly show the readers about how data flows from remote server and from cache interface system based on the basis of the keys set in the control file. The 'getRemote' is the call to remote server to get the response object which will be set to cache system object through the 'setCacheObject' method.

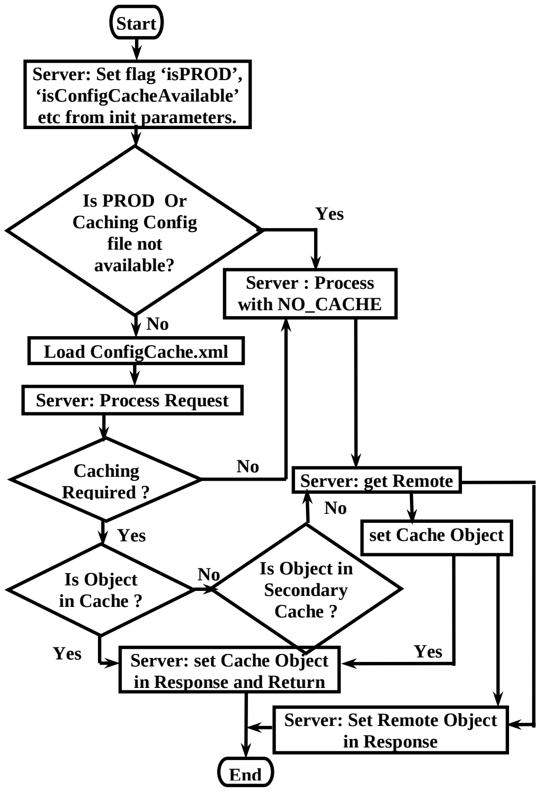

Start
Server: Set flag 'isPROD', 'isConfigCacheAvailable' etc from init parameters.
Is PROD Or Caching Config file not available?
Yes
Server : Process with NO_CACHE
No
Load ConfigCache.xml
Server: Process Request
Caching Required ?
No
Server: get Remote
No
set Cache Object
Yes
Is Object in Cache ?
No
Is Object in Secondary Cache ?
Yes
Server: set Cache Object in Response and Return
Yes
Server: Set Remote Object in Response
End

Figure 7. Application Data Flow in Cache Framework

The flow diagram(refer figure 7) depicts the detailed flow, how the primary cache system and the secondary cache system works out in the cached enabled framework, and how the PRODUCTION environment works as if the system works with NO_CACHE configuration mechanism.

8. PERFORMANCE ANALYSIS

Here, we will visualize mathematically how this framework approach is more efficient in development environment than the traditional development approach.

Let's consider following symbols with the respective description.

δ_r be the time required to process the request object in the server, δ_f be the time required to process and load the cache config file.

However δ_f is very small amount time, $\delta_f \rightarrow \in$ and $\in \rightarrow 0$ in advanced systems and servers with high configuration. δ_s be the time required to set or reset the response object in the server, δ_t be the transmission time for fetching the data from the remote server and δ_l be the time required to load from cache.

So in traditional development approach, the time required to process n - number of request by the server is given by

$$\Delta_c = \delta_f + [\delta_r + \delta_s + \delta_t] + \sum_{i=2}^{n} [\delta_r + \delta_s + \delta_l]$$

$$= [\delta_f + \delta_t] + n[\delta_r + \delta_s] + (n-1)\delta_l$$

In case of the proposed Data Caching approach, the time required to process same n-number of request by the server is given by

$$\Delta_d = \sum_{i=1}^{n} [\delta_r + \delta_s + \delta_t]$$

$$= n[\delta_r + \delta_s + \delta_t]$$

$$= \delta_f + [\delta_r + \delta_s + \delta_t] + (n-1)[\delta_r + \delta_s + \delta_l]$$

$$= \delta_f + [\delta_r + \delta_s + \delta_t] + (n-1)[\delta_r + \delta_s + \delta_l]$$

The difference between the traditional development and the proposed caching approach is given by

$$\Delta = \Delta_d - \Delta_c$$

$$= n[\delta_r + \delta_s + \delta_t] - [[\delta_f + \delta_t] + n[\delta_r + \delta_s] + (n-1)\delta_l]$$

$$= n[\delta_t] - [\delta_f + \delta_t] - (n-1)\delta_l$$

$$= (n-1)[\delta_t] - (n-1)\delta_l - [\delta_f]$$

$$= (n-1)[\delta_t - \delta_l] - [\delta_f]$$

As we have already identified the local ConfigCache.xml load and process time is very minimal($\delta_f \rightarrow 0$) for advanced systems and servers. So; $\Delta = (n-1)[\delta_t - \delta_l]$. i.e. $\Delta \geq 0$ as $\delta_t > \delta_l$, and $n \geq 1$

If we take a close look, this approach will work where there is a more than one request in the system. For $n = 1; \Delta = 0$; which shows that the response time in traditional approach and in data caching approach is same.

In the system where $n > 1; \Delta > 0$ as $\delta_t > \delta_l$
So this approach will perfectly suitable where the development environment sends more than one requests and the system will be more efficient when the number of request increases.

This framework is expected to perform extremely well in a middle ware service systems e.g. web services, Enterprise Java Beans applications, MQ services etc, where the system is accessed by more users or systems and is receiving more number of requests.

9. DATA SIMULATION ANALYSIS

A sample simulation of this framework implemented on a two tiered architecture based application. Around 50 times hit is taken into consideration with similar and different type of requests in both approach, and the first 10 sample results are as shown below.

Here in the below table(refer Table 1), we can see the caching framework take a slight more time in the first hit, this is due to the load the control files and setting up the environment with the required flags as explained in the previous sections.

Table 1. Response time for Traditional Vs Proposed data caching framework

HIT #	TRADITIONAL(MSEC)	NEW(MSEC)	DIFF(Δ)
1	33172	35172	-2000
2	31266	31219	47
3	35766	0	35766
4	33759	08	33751
5	32143	31469	674
6	33579	0	33579
7	32918	03	32915
8	35611	35421	190
9	32453	07	32446
10	35746	0	35746

We have hit other external interface system (remote server) with random set of data with NO_CACHE traditional system Vs WITH_CACHE framework system. The system with data caching framework reduces the response time as in the expected way.

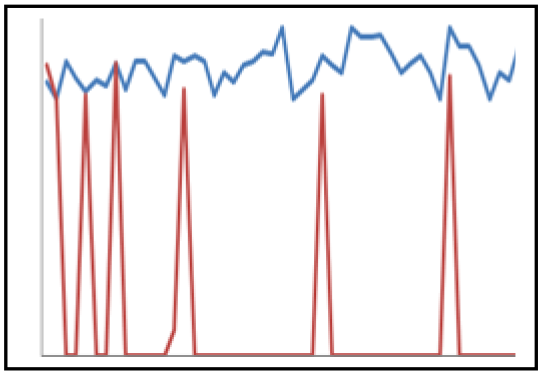


Figure 8. Response Time for Traditional and Data Caching

Figure 8 shows the hit sequence Vs the response time for the both of the approach. The upper zigzag line shows the traditional approach response time where as the lower spikes are for the data caching approach. At the spike, the data caching approach has hit the remote system with the new data, or the cache system does not have the data in its cache.

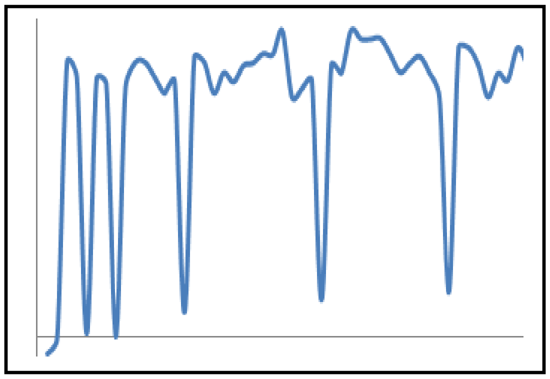


Figure 9. Performance for random request in Traditional Vs Data Caching

The above picture (figure 9) shows the hits Vs the difference (Δ) which depicts that the data caching framework is in below 0-level in the first hit, and the Δ is always in positive gain. The downwards spikes provide the information that the application hit the remote system when the requested data not found in the cache. Though it sometimes has some downwards spikes, but still in development phase, it has the performance gain over the traditional approach. Figure 10 shows that for every hit, the data found in cache system. These scenarios occurred when the entire set of request processed initially and then the system gets the same request for successive fashion, and this is a perfect scenario when we intend to test the application many times in the development phase.

The first time hit starts with below 0-level, as it takes a little more time to load the frame in the environment and there onwards it has a significant improvement of performance over the proposed cache framework over the traditional development approach.

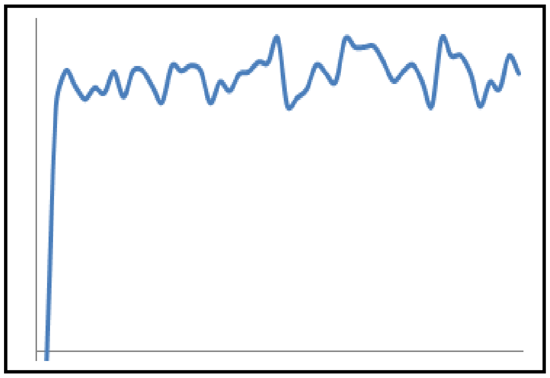


Figure 10. Performance for Cached items in Traditional Vs Data Caching

This framework is implemented in List, and then in Hash map and currently in Tree map. A sample scenario of 1000 requests is simulated against the response time in milliseconds.

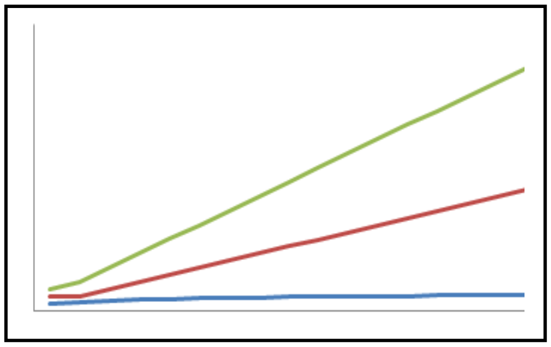


Figure 11. Performance measure List Vs Hash Map Vs Hash Tree

The above figure (figure 11) shows the gradual performance improvement of the framework through the implementation of different data structure model in different versions. The Tree Hash Map implementation has the lowest response time than the List and Hash Map implementation.

10. CONCLUSION

The data caching framework approach proposed in this paper has a better performance over the traditional development process. In the above section, the performance is compared and found that the code and unit test time could be minimized significantly for an application in the development phase. The proposed framework is well suited in two tier and three tier based applications where each tier, the proposed framework could be configured. The data security concerns are taken care of in this proposed approach as this framework does not require a persistent storage base e.g. file system, database etc, for caching the data objects. As we have seen in the section 3, there is no additional infrastructure required to establish this frame work. So this approach could be configured perfectly where the data server is located remote location and the application is very slow in its development phase, when it is expected to be developed in other geographical location.

This proposed framework could be extended to implement the more cache layers so that it would fetch the data from inner level of the cache objects, if the primary or secondary layer cache fails. In n-layer cache implementation the feasibility of the number-n is to be verified against the real data response time. The cache object indexing, implementations of advanced cache clustering and buffering techniques are the other areas of this proposed frame work in its future scope. Currently the request-response pattern study to resolve the bulk data processing issues, and the more advanced data structure model e.g. B^+ tree, B^* tree, AVL tree implementations are in progress which would tune the search process to provide a better performance. This framework is currently under study for designing in wireless sensor network to route the network packets which save the response time and the battery consumption.

REFRENCES:

[1] Q. Ren, and M. H. Dunham, "Semantic Caching and Query Processing", *Transactions on Knowledge and Data Engineering* , 2003, pp. 192-210.

[2] Zheng, B., Xu, J., and Lee, D, "Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments", *IEEE Transactions on Computers*, 2002, pp. 1141–1153.

[3] Q. Ren, and M. H. Dunham, "Using semantic caching to manage location dependent data in mobile computing", Proceedings of the 6th annual international conference on Mobile computing and networking, 2000, pp. 210-221.

[4] Q. Ren, and M. H. Dunham, "Using clustering for effective management of a semantic cache in mobile computing", Proceedings of the 1st ACM International Workshop on Data Engineering for Wireless and Mobile Access, 1999, pp. 94-101.

[5] A. M Keller, and J Basu, "A predicate-based caching scheme for client-server database architectures", The VLDB Journal, 1996, pp. 35-47.

[6] 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.

[7] A. J. Smith, "Cache Memories", ACM Computing Surveys, 1982, pp. 473-530.

[8] P. J. Denning and S. C. Schwartz, "Properties of the Working-set Model", Communications of the ACM, 1972, pp. 191-198.

[9] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems", IEEE Trans. Computers Vol.5, 2002, pp. 608-621.

[10] S. Evron, "A Practical Guide to Data Caching with Zend Server", White Paper On Zend, 2009, pp. 1-11

[11] P. Godfrey and J Gryz, "Answering queries by semantic caches", Proceedings of the 10th International Conference, DEXA volume 1677 of LNCS, 2009, pp. 485-498

[12] K. C. K. Lee, H. V. Leong and A. Si, "Semantic query caching in a mobile environment", ACMSIGMOBILEMobile Computing and Communications Review, 1999, pp. 28-36

[13] G. Liu, A. Marlevi and G. Maguire, "A mobile virtual-distributed system architecture for supporting wireless mobile computing and communications", Wireless Networks, 1996, pp. 77-86

[14] J. Peissig, "guidePort – An Information and Guidance System", Wireless Protocols for Network Communication Proceedings, 2004, pp. 1-17

[15] M. Franceschetti, M. D. Migliore, and P. Minero, "The degrees of freedom of wireless networks", Information theoretic and physical limits. In Proc. Allerton Conference.2003

[16] U. Niesen, P. Gupta, and D. Shah, "The Balanced Unicast and Multicast Capacity Regions of Large Wireless Networks", IEEE Transactions on Information Theory vol. 56, 2010, pp. 2249-2271

[17] P. D. R. Vijayakumar and T. Ravichandran, "Cooperative Caching Protocol for Multimedia Data in Mobile Ad Hoc Networks", European Journal of Scientific Research, 2011, pp. 392-403

[18] U. Niesen, "Caching in Wireless Networks", IEEE Transactions on Information Theory, 2012, pp. 6524-6540.