IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-10, Oct 2018 E-ISSN: 2347-2693

Analysis and Performance of Cache Using Persisted Java Topics

Ankush Sharma[1*] , Vishal Gupta[2]

[1,2]M.C.A, Model Institute of Engineering And Technology, Jammu, India

[*]*Corresponding Author: Ankush.mca@mietjammu.in,*

Available online at: www.ijcseonline.org

Accepted: 15/Oct/2018, Published: 31/Oct/2018

Abstract: In most of the applications in today's world, data is fetched from the secondary storage i.e. hard disk. The user connects directly to the database for fetching the data. In current approach of cache management and loading, data is loaded from the secondary storage i.e. from the database to the primary storage i.e. cache. In case of any cache failure, entire data has to be reloaded from the database which consumes a lot of time in case the volume of data runs into millions. In the proposed design, data will be fetched from cached data and will be displayed to the end user. In this design of the cache reload we will be persisting the cached objects in some persistent storage as jms topics or flat files. The cache will be rebuilt again from these objects rather than from the database in case of any cache failures. In addition to it, analysis and performance of cache has been shown in this paper considering various parameters.

Keywords: Coherence cache, hard drive, secondary storage

I. INTRODUCTION

In the Present software applications design, there are various requests made to the Database from several applications like ADF, SOA etc. These database calls make resources busy between applications and data base. Frequently database call makes system/Applications slow. In present scenario, we are dealing with large applications with loads of read and write tasks and every call made to the data base will cost more from the performance point of view. As a workaround, we will provide solutions cache, a cache is a component that transparently stores data so that future request for that data can be served faster. There are many cache management/No SQL products such as Couch base, Hazelcast, and Coherence etc. available in the market. In our study we will be using Oracle Coherence.

Now as the cache is not persistent therefore in case of any problem with the cache such as Cache invalidation, Cache server failure etc. we will be again required to build the cache from Database which will be a time taking process in case the data we are loading in the cache store is highly de-normalized and millions of records required to be loaded in the cache. So in our study we will try to find a solution to persistent the cached java objects in persistent storage such as flat file or JMS topic so that in case a rebuild of cache required it can then be done with each from this storage rather than going for the DB calls.

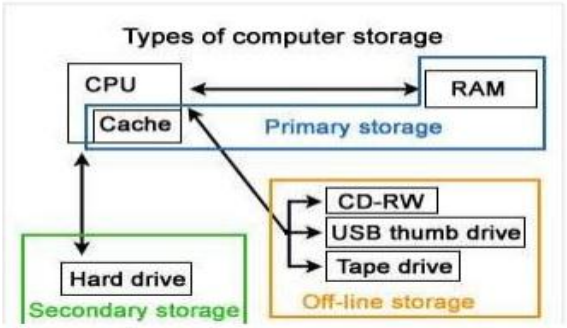

Fig 1.1 Secondary storage

A. Coherence Cache

Coherence is the industry prominent in memory data lattice that permits to certainly balance the assignment critical applications by providing the fast admittance to data .As data sizes are growing day by day in the whole world, performance and scalability becomes the key for an applications success or failure. Oracle coherence provides the capability to store the data in primary memory as shown in fig 1.1 as named caches. Each cache can contain the de-normalized data from multiple tables so as to have easy access to same in different named caches. The named caches can be maintained on logical or functional division of the application whichever applicable.

Below fig shows data to be loaded from Secondary storage i.e. database to primary storage i.e. cache. Then data from this cache is accessed by the application.

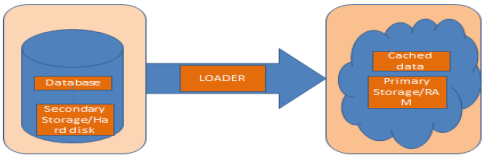

Fig. 2

II. CURRENT APPROACH

In most of the applications in today's world, data is fetched from the secondary storage i.e. hard disk and user connects directly to the database for fetching the data . Below sequence diagram shows the conventional dataflow in most of the applications today.

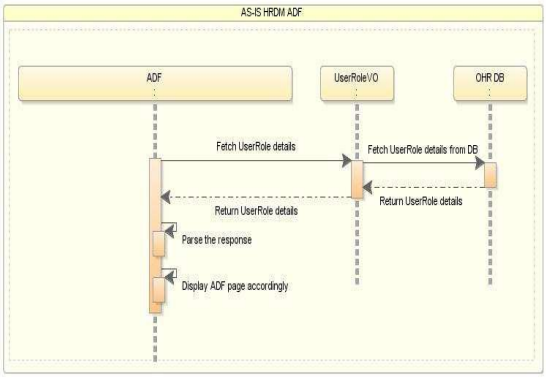

Fig. 3

III. PROPOSED APPROACH

In the new design, data will be fetched from cached data and will be displayed to the end user. Below is the sequence diagram for the same.

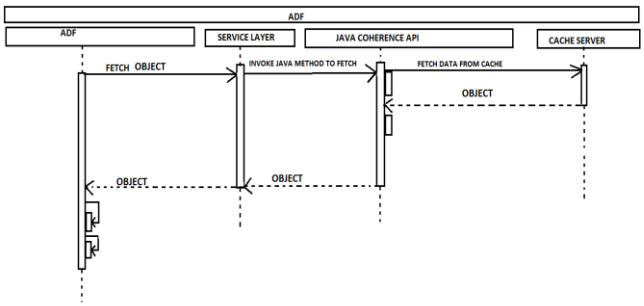

Fig. 4

IV. CURRENT APPROACH OF CACHE LOADING AND MANAGEMENT

In current approach of cache management and loading, data is loaded from the secondary storage i.e. from the database to the primary storage i.e. cache .Below diagram shows the process.

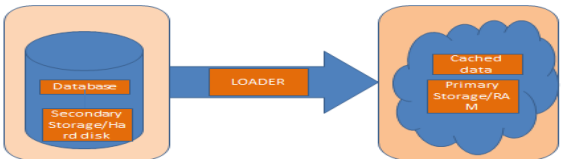

Fig.5

However in case of any cache failure, data has again to be loaded from the database all over again which is a painful area again in case the volume of data runs into millions and if no of tables involved are say 3-40 then amount of time required to load this data all over again to the cache is very costly and time taking process which can even sometimes take upto 24 hours. Thus making the application unavailable for use for the said period . In second part of our study we will be trying to find a solution to enable the faster reloading of the cache in case of its failure. Next section explains the same

V. PROPOSED DESIGN OF CACHE RELOAD

In proposed design of the cache reload we will be persisting the cached objects in some persistent storage as jms topics or flat files and in case of cache failures,the cache will be rebuild again from these objects rather than from the database.

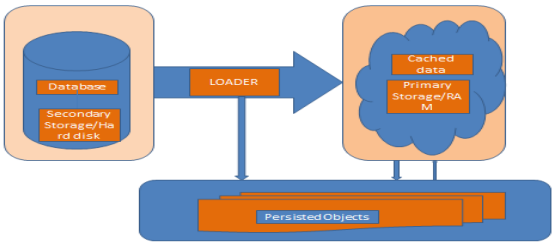

Fig. 6

VI. LITERATURE REVIEW

Kuber Vikram Singh et al. [1] proposed, applications like ADF saves lot of time as these applications don't fetches the data from secondary memory. They proposed batch process schedule for transferring data from secondary memory to primary memory. For their research they used java Regex for pattern generation and matching the suspicious attack signatures

Tamura et al. [2] proposed, several designs that load and lock cache via hardware techniques in order to load every single block. The main disadvantage is that it demands variations both in cache memory and main memory organisations

Dr. Vivek Chaplot et al., [3] evaluated performance of cache memory by using various factors like cache access

time, miss rate and miss penalty. The performance of cache memory is much better as compare to RAM as it has faster access time. The main disadvantage of cache memory is the on-chip energy consumption

Sharon Mathew et al., [4] proposed, a unified approach for process scheduling and L2 cache partitioning in order to increase the performance of embedded applications.

Ronak Patel et al., [5] observed that SRAM and STT-RAM are about equally fast for performing read operations, however STT-RAM takes longer to perform write operations and therefore also consumes more energy for write operations. SRAM and STT-RAM were found as very common designs used for building cache memory. However, there are several companies and organizations doing research on how to improve write performance and efficiency on the STT-RAM. Response analyzer and Modifier module deals with the data to be returned the client, it modifies the malicious response to harmless data. Attack Recorder and Response Rejecter Module records the malicious Request/Response for future use. Java Regex has been used for pattern generation and matching the malicious attack signatures.

Dimple et al. [6] proposed an ant based framework to balance the load. In their research, the author proposed an active ant at both client side and the server side. The client ant is responsible for the user request whereas the server ant is responsible for replying the request. The authors improved the server performance in their research.

Sandeep et al., [7] presented the performance analysis of static and dynamic load balancing algorithms. Comparison is done on the number of parameters such as overload rejection, fault tolerance, accuracy and stability etc. Load balancing algorithm is selected on the basis of situation in which work load is assigned i.e. at run time or compile time. Dynamic load balancing algorithms are proved to be less stable as compare to static algorithms.

Joseph et al., [8] examined that client-side caching is complementary to data scheduling in improving the performance of real time information dispatch systems. An effective caching mechanism retains data items that are most likely to be accessed by clients and reduces the number of requests submitted to the server over the wireless communication channel. This saves the narrow bandwidth and reduces the workload on the server; also it helps in reducing access latency by serving requests locally with data cached at the clients.

Hossam et al., [9] Web caching is a popular technique to improve the performance and scalability of the Web by increasing document availability and enabling download sharing. Using cache cooperation, a mechanism for sharing documents among caches can improve performance of the system. Further, it can improve performance by providing a shared cache to a large user population.

Liu et al., [10] proposed that an adaptive technique is used for reducing Web traffic and to access the Web sites efficiently. The proposed algorithms at client side and server side are efficient to reduce the Web traffic in adaptive manner. Since it is a hybrid technique, latency is reduced to 20 – 60 % and cache hit ratio is increased 40 – 82 %.

VII. RESEARCH METHODOLOGY

Tools and servers installed and configured for this research

A. Oracle

Installed oracle 11g express edition .Now the installed oracle database server is running on machine as service. Created table employee required for this research using oracle sql commands.

Below is the SQL statement used to create this table

```
CREATE TABLE "HR"."EMPLOYEES"
(  "EMPLOYEE_ID" NUMBER(6,0),
"FIRST_NAME" VARCHAR2(20 BYTE),
"LAST_NAME" VARCHAR2(25 BYTE)
CONSTRAINT "EMP_LAST_NAME_NN" NOT
NULL ENABLE,
"EMAIL" VARCHAR2(25 BYTE)
CONSTRAINT "EMP_EMAIL_NN" NOT NULL
ENABLE,
"PHONE_NUMBER" VARCHAR2(20 BYTE),
"HIRE_DATE" DATE CONSTRAINT
"EMP_HIRE_DATE_NN" NOT NULL ENABLE,
"JOB_ID" VARCHAR2(10 BYTE)
CONSTRAINT "EMP_JOB_NN" NOT NULL
ENABLE,
"SALARY" NUMBER(8,2),
"COMMISSION_PCT" NUMBER(2,2),
"MANAGER_ID" NUMBER(6,0),
"DEPARTMENT_ID" NUMBER(4,0),
CONSTRAINT "EMP_SALARY_MIN" CHECK
(salary > 0) ENABLE,
CONSTRAINT "EMP_EMAIL_UK" UNIQUE
("EMAIL")
USING INDEX PCTFREE 10 INITRANS 2
MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST
GROUPS 1 BUFFER_POOL DEFAULT
FLASH_CACHE DEFAULT
CELL_FLASH_CACHE DEFAULT)
```



```
        TABLESPACE "USERS"  ENABLE,
        CONSTRAINT "EMP_EMP_ID_PK"
        PRIMARY KEY ("EMPLOYEE_ID")
        USING INDEX PCTFREE 10 INITRANS 2
        MAXTRANS 255 COMPUTE STATISTICS
        STORAGE(INITIAL 65536 NEXT 1048576
        MINEXTENTS 1 MAXEXTENTS
        2147483645
        PCTINCREASE 0 FREELISTS 1 FREELIST
        GROUPS 1 BUFFER_POOL DEFAULT
        FLASH_CACHE DEFAULT
        CELL_FLASH_CACHE DEFAULT)
        TABLESPACE "USERS"  ENABLE,
        CONSTRAINT "EMP_DEPT_FK" FOREIGN
        KEY ("DEPARTMENT_ID")
        REFERENCES "HR"."DEPARTMENTS"
        ("DEPARTMENT_ID") ENABLE,
        CONSTRAINT "EMP_JOB_FK" FOREIGN
        KEY ("JOB_ID")
        REFERENCES "HR"."JOBS" ("JOB_ID")
        ENABLE,
        CONSTRAINT "EMP_MANAGER_FK"
        FOREIGN KEY ("MANAGER_ID")
        REFERENCES "HR"."EMPLOYEES"
        ("EMPLOYEE_ID") ENABLE
)
```

Entered data in this table required for this research using below sql queries

```
        Insert into EMPLOYEES
        (EMPLOYEE_ID,FIRST_NAME,LAST_NA
        ME,EMAIL,PHONE_NUMBER,HIRE_DAT
        E,JOB_ID,SALARY,COMMISSION_PCT,M
        ANAGER_ID,DEPARTMENT_ID) values
        (100,'Steven','King','SKING','515.123.4567',to
        _date('17-06-03','DD-MM-
        RR'),'AD_PRES',24000,null,null,90);
Insert into EMPLOYEES
(EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMA
IL,PHONE_NUMBER,HIRE_DATE,JOB_ID,SALAR
Y,COMMISSION_PCT,MANAGER_ID,DEPARTME
NT_ID) values
(101,'Neena','Kochhar','NKOCHHAR','515.123.4568',to
_date('21-09-05','DD-MM-
RR'),'AD_VP',17000,null,100,90);
```

Created a java class to load data from database and create a array of objects of the same data.The objects created will be of java class . In this, we have used the following class

```
package com.model;

import java.io.Serializable;

public class Employee implements Serializable{

        String empid;
        public String getEmpid() {
                return empid;
        }
        public void setEmpid(String empid) {
                this.empid = empid;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public int getAge() {
                return age;
        }
        public void setAge(int age) {
                this.age = age;
        }
        String name;
        int age;

}
```

Below is the code which fetch from database and maps to the object of the above mentioned class

```
public  HashMap getCachemap(Connection con) throws
SQLException {
                Statement st=con.createStatement();
                ResultSet  rs=st.executeQuery("Select  *
from employees");
                HashMap cachemap=new HashMap();
                List ls=new ArrayList();
                while(rs.next())
                {
                        ls.add(rs.getString(1));
                        Employee          empl=new
Employee();
                        empl.setEmpid(rs.getString(1));
                        empl.setName(rs.getString(2));
                        cachemap.put(rs.getString(1),
empl);
                        System.out.println("Employee
name is "+empl.getName());
                }
                st=null;
                return cachemap;
        }
```

B. WebLogic Server

Installed WebLogic server 12c and configured the same.

In this module we have created one connection factory and one jms durable topic.

C. Connection Factory

JMS Connection factories can configure properties of the connections returned to the JMS client, and also provide configurable options for default delivery, transaction, and message flow control parameters.

D. JMS TOPIC

Characterizes a publish/subscribe end type, which are utilized to non-concurrent companion interchanges. A message delivered to a topic is distributed to all topic consumers. Several aspects of a topic's behaviour can be configured, including thresholds, logging, delivery overrides, delivery failure, and multicasting parameters. Created a program which will publish the objects onto the above mentioned jms topics .These are the same objects which are read from the database as mentioned above .Now these objects are published to the created JMS durable topic which is created in the above step .

E. Coherence Cache server

Coherence cache server will be used to build cache store. The cache server will have named cache which is nothing but a hash map. In this named cache the java objects will be store in key value pair. The object will be read using key and the other CRUD operations will be done using key .First the cache in this cache server will be loaded from oracle RDBMS .the a client is develop to connect to this cache and read from cache so as to demonstrate the application performance improvements by using cache. Second, this cache server will be restarted. Once restarted all the cache in this cache server will be lost. Then a code is developed which will load cache into this cache server from the persisted objects on jms topics .In this step we will also demonstrate the loading performance improvement when done from jms topic.

VIII. RESULTS

The volume of data with which we have tested when we load data from database, it took about 104(ms) and when done from java objects persisted on java topics, it took about 62(ms).Search speed also gets increased when searching is done from persisted java topics as compared to when done from database. It took around 79(ms) when done from JMS topics and 122(ms) when done from database named Employee. Results shown in table (I).

TABLE I. COMPARISON TABLE OF PROPOSED TECHNIQUE

Characteristics	Cache reload from database named Employees	Cache reload from persisted java topic
Performances	Good	Best
Search Speed(ms)	122ms	79ms
Time(ms) to reload cache	104ms	62ms

TABLE II :COMPARISON TABLE OF PROPOSED TECHNIQUE

Characteristics	Cache reload from database named Employees	Cache reload from persisted java topic
Performance	Good	Best
Search Speed(ms)	122ms	79ms
Time(ms) to reload cache	104ms	62ms

TABLE III: Performance parameter for database named Employees (Speed and Time)

Volume Of data	Search speed (mS)	Time to reload cache (mS)
1.0GB	122	104
1.5GB	141	112
2.0GB	177	128
2.5GB	185	132
3.0GB	210	149

TABLE IV: Performance parameter For Persisted java Topics (Speed and Time)

Volume Of data	Search speed(ms)	Time to reload cache (ms)
1.0GB	79	62
1.5GB	85	69
2.0GB	93	78
2.5GB	115	89
3.0GB	129	98

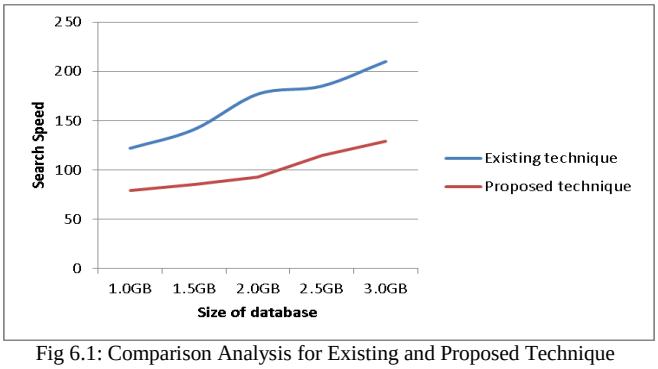

Fig 6.1: Comparison Analysis for Existing and Proposed Technique (Search Speed vs Size of database)

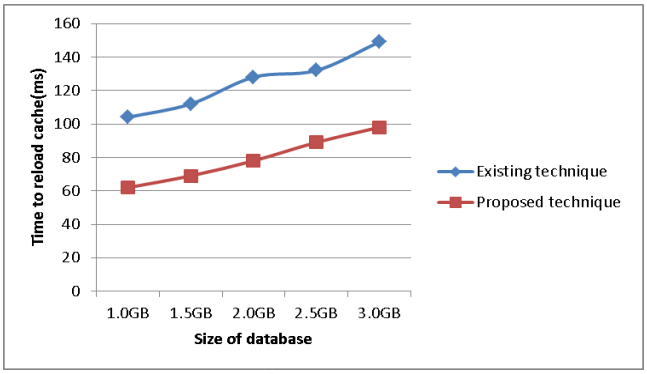

Fig 6.2: Comparison Analysis for Existing and Proposed Technique (Time to reload vs Size of database)

TABLE V: Performance parameter For database (Speed and Time)

Number Of Records	Search Speed (ms)	Time to reload Cache
100	122	104
200	133	106
300	141	107
400	153	109
500	166	111

TABLE VI: Performance parameter For Persisted java Topics (Speed and Time)

Number Of Records	Search Speed (ms)	Time to reload Cache
100	79	62
200	85	64
300	91	65
400	97	68
500	104	71

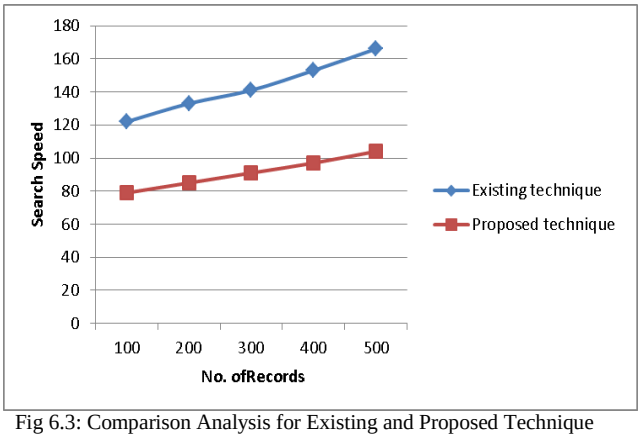

Fig 6.3: Comparison Analysis for Existing and Proposed Technique (Search Speed vs No. of records)

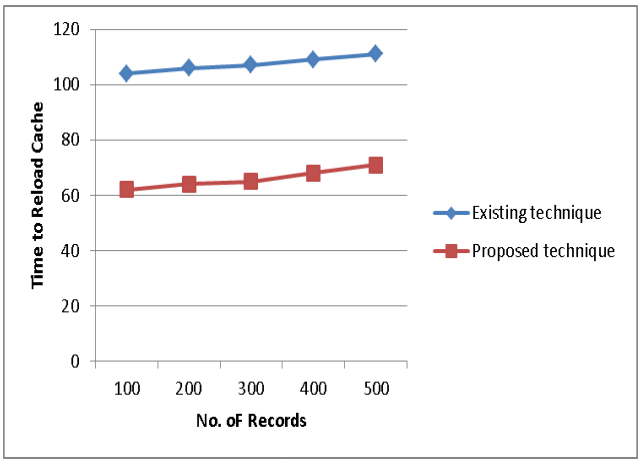

Fig 6.4: Comparison Analysis for Existing and Proposed Technique (Time to reload cache vs No. of records)

IX Conclusion

Hence from above analysis, it is proved that loading java objects from persisted java topics, we get better results as compared to loading data from database. It is a significant technique to load cache from persisted java topics rather than loading cache from database.

References

[1] Kuber Vikram Singh and Inderjeet Yadav, "Improving Data Access Performance Using Coherence Caching in SOA and ADF Application." International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 3 Issue: 5 (2015)

[2] Tamura, E., et al. "High performance memory architectures with dynamic locking cache for real-time systems." Proceedings of the 16th Euromicro Conference on Real-Time Systems, Italy. 2004.

[3] Chaplot, Vivek. "Cache Memory: An Analysis on Performance Issues." International Journal 4, no. 7 (2016).

[4] Mathew, Sharon, and D. M. Jagadeeswari. "Task scheduling and memory partitioning for multiprocessor system-on-chip using low-power L2 cache architecture." Int J Emerg Trends Eng Dev 2, no. 3 (2013): 95-105.

[5] Patel, Ronak. "Implementation of Cache Designs and Efficiency of Cache Memory."

[6] Dimple Juneja and Atul Garg, "Collective Intelligence based framework for load balancing of Web servers", IJICT, Vol 3 No-1 Jan-2012

[7] Sandeep Sharma, S.Singh and Meenakshi, "Performance analysis of load balancing algorithms", World academy of science, 2008

[8] Jospeh Kee Yin Ng and Chui Ying Hui, " Client-Side caching strategies and On demand broadcast Algorithms for Real Time Information", IEEE , March 2008.

[9] Hossam Hassanein, Zhengang Liang and Patrick Martin, "Performance Comparison of Alternative Web Caching Techniques", Procceedings of the seventh International Symposium on Computers and Communications, IEEE, 2002.

[10] M. Liu, F. Wang, D. Zeng and L.Yang, "An Overview of world wide Web Caching", International conference on Systems Man and Cybernetics, IEEE, 2001, pp.3045-3050.

