Improving Performance of Cloud based Transactional Applications using In-Memory Data Grid

Indu Arora
Department of Computer Science and Applications
MCM DAV College for Women
Chandigarh-India

Anu Gupta, Ph.D.
Department of Computer Science and Applications
Panjab University
Chandigarh-India

ABSTRACT
Browser based applications are used currently to handle data related requirements of educational institutes. Such applications are not scalable due to limited scalability of database servers. The performance of such applications can be improved with the use of Cloud Computing and In-memory Data Grid (IMDG). IMDG uses the concept of caching to keep frequently used data in memory which is required by an application. This ensures high availability of data to the application. Due to cached data, the performance of the application also increases. This paper proposes the usage of IMDG for deploying transactional applications of educational institutes in the Cloud. This paper also points out performance issues in using IMDG. Then the paper proposes an approach to solve the identified performance issues of transactional applications required by educational institutes in Cloud environment. An analysis of proposed approach with traditional approach highlights better access time, availability and scalability.

General Terms
Performance Analysis

Keywords
In-Memory Data Grid, Transactional Applications, Cloud Computing, Educational Institutes.

1. INTRODUCTION
The educational institutes require both analytical and transactional applications to improve the quality of education. Traditional Client/Server or Browser based applications are used to meet the data related requirements of educational institutes. In traditional application architecture, there exists a tight coupling of business layer with database layer [1]. This means business layer knows the location of database layer. The distribution of database layer in Cloud environment changes the data access procedure of the application. Therefore, some traditional applications cannot be used in Cloud as such. However, the factors like speed, scalability, fast deployment, cost control, reliability and security are responsible for moving applications to Cloud [2]. But transactional applications cannot be deployed in the Cloud due to eventual consistency. Data grid may play an important role in deploying applications in Cloud [3]. Data Grid often keeps data fully or partially as data sets. Cloud Computing and In-memory Data Grid (IMDG) can be used together to deploy transactional applications in Cloud. They can play important role in improving performance of transactional applications for educational institutes while ensuring scalability.

The paper has been structured into six sections. Second section details In-memory Data Grid and its usage. Third section discusses performance issues in using In-memory Data Grid. Fourth section proposes an approach to handle these issues. Fifth Section gives an insight into the performance issues with experimentation and case study of an application of educational institutes. Sixth Section analyses the performances of traditional method to read data and reading through In-memory Data Grid followed by conclusion and scope for future work.

2. IN-MEMORY DATA GRID
Cloud environment based applications use various strategies for accessing data from cloud like Web Services, and In-Memory Data Grid [4]. In-Memory Data Grid (IMDG) stores large volume of data entirely in memory distributed across servers. These servers are connected to each other in a cluster. This is possible due to availability of advanced technologies such as 64-bit processor, multi core processers, large memory size, better networking and fast communication technology. Some popular available IMDG tools are Oracle Coherence, Gemfire, Gigaspace, Gridgrain, eXtreme Scale. The primary purpose of using IMDG is to keep frequently used data required by an application in memory. This ensures high availability of data to applications. Such applications do not need to interact with the database directly, so it reduces time involved in expensive disk reads. IMDG technology is used in data processing applications like high energy Physics data and medical imaging, as it significantly reduces the time and cost to obtain desired results [5][6].

IMDG requires large memory size to process data. If sufficient memory is not available, system will automatically swap pages [7]. With this, benefits of caching will fade away. Since, main memory is a limited resource as compared to disk, so IMDG uses horizontal scalability to add more nodes on-demand in real-time. In addition to horizontal scaling, it also supports vertical scaling. While in horizontal scaling, servers are added or reduced in the cluster depending upon requirements, vertical scaling increases the processing capability of servers by increasing RAM etc. Large-scale Web applications require better performance. So, there has been a surge in the use of DRAM. Both Google and Yahoo! store their search indices entirely in DRAM [8].

Data is usually stored in terms of relational structures for transactional applications. But IMDG stores data as a pair of key and value. The data as object is identified uniquely in a cluster by a key. In IMDG, Data partitioning is done automatically and then partitioned data is uniformly stored across servers in clusters to ensure better availability of data in case of a failure of any server. One server contains primary data and copy of same data on another server is known as backup data. Depending upon availability of servers, backup partition can be stored on more than one server. IMDG does not allow single point of failure in the application by distributing its objects and related processing across multiple physical servers [9]. The In-memory data is very close to applications and thus reduces access time. Oracle coherence, In-memory Data Grid moves data closer to applications to

reduce latency and improve performance [10]. IMDG supports thousands of updates per second on the data stored in cache/memory. Hence it increases the performance of applications. This technology can be used in developing transactional applications for the Cloud. Cloud based memory architecture for High-Performance Computing (HPC) applications in Cloud enhances performance of HPC applications and utilizes Cloud computing resources better [11].

3. PERFORMANCE ISSUES IN USING IMDG

For improved performance, a transactional application developed using IMDG is required to access data from cache/memory. Such features are available in languages like Java. Java Persistence APIs (JPA 2.0) provide concept of shared cache [12]. There are two types of caching: Object caching and Data Caching. Object Caching includes data, structure and relationships, while Data caching stores objects having database rows only. Object Caching based framework is being used for improving data access performance in an enterprise application [13]. One of the advantages of using data cache is that there is no need to track relationships. Some JPA providers support separate caching for relationships that store only Object ID for reference to avoid stale data. During cache hit, if data is not available in the cache, the database is searched. This degrades the required performance. There are two types of caching: Table level and Query level [14]. Table level caching stores entire table data in cache. Query level caching stores result of a query. Different types of caching strategies like Full, Weak and Soft are used in application architecture in Cloud environment [15]. Full caching never flushes data until deleted explicitly. Weak caching keeps data till application references exist and data has not been garbage collected. Soft caching is similar to weak caching but uses garbage collection hints to release data whenever memory is low.

It is not possible to store entire data at a time in the memory of a single machine. Therefore, data is stored in memory across the servers in distributed manner. Transactional applications require write operations more frequently on data stored in cache across servers. Such transactional applications need to maintain ACID (Atomicity, Consistency, Integration, Durability) guarantees which is a major challenge while deploying them in the Cloud.

It is also very challenging to ensure performance of applications in terms of high availability and reliability besides providing ACID support especially when data is stored in multiple servers in a cluster, IMDG tool and other middleware. Whenever a transactional application server is started, it requires a part or whole of master data to be in the cache. Putting such large data in cache degrades performance initially. After some time, data lying in the cache becomes stale or inconsistent especially when one application is accessing data from the cache/memory and the other application or part of the same application updates data directly on the database. Data meant for read only purpose poses no issue. But when insert, delete and update operations are to be performed, there is high possibility of data becoming stale. Consider another scenario, in which data is stored in the cache/memory of multiple servers in a cluster and update takes place only at one location, then same data stored on other locations becomes out of date. These are the main issues and challenges that need to be addressed during development of transactional applications accessing data from cache/memory.

4. PROPOSED APPROACH TO RESOLVE PERFORMANCE ISSUES

Scalability, fast data retrieval and high availability become major issues with Browser based application software. Such applications generally rely on a single database server. So these applications are not scalable as they support scaling of application server but not of database server. Database tier is complex to replicate and may suffer from the problem of data synchronization and consistency [16]. Cloud computing is able to provide scalability, reliability and high availability of data. It has been observed that read-mostly analytical data management applications are better suited for deployment in the Cloud than transactional data management applications [17]. Clouds can be used with distributed database for handling very large databases maintaining availability, scalability as well as reliability [18]. Cloud computing provides eventual consistency which is not considered suitable for deploying transactional applications in the Cloud as the basic requirement of transactional applications is to ensure ACID guarantees.

The proposed approach is to use any middleware which supports IMDG to ensure ACID guarantees while deploying transactional applications in the Cloud. IMDG tools also provide other services like failover recovery and backup but persistence (writing to database) is still a problem. On the other hand tools like CloudTran, extends the capability of IMDG like Oracle Coherence and Gigaspaces to provide high speed and ACID compliance transactions across different servers. The usage of IMDG tools and middleware like CloudTran provides scalable, reliable and faster access to data. The proposed solution is described in Figure 1.

Performance of transactional applications can be achieved at three levels during development and deployment of transactional applications. At database level, optimization techniques like indexing, primary key, references etc. are used. At hardware level, high end servers are required. At application level, tools and techniques like IMDG, Oracle Coherence, CloudTran and Java can be used [19-22]. The optimization at hardware and database level is beyond the scope of this paper. The focus of this paper is on optimization at the application level. In view of the above mentioned performance issues, solution to resolve them is proposed and verified by implementing an application of Student Registration Return System (SRRS) for educational institutes. The solution to each of the performance issues is proposed below.

4.1 Initial Data

In the proposed solution, two-phase approach is used to resolve issue of Initial Data: Filtering and Loading. First phase of Filtering includes the filtering of data based on some criteria. In the second phase, the filtered data is loaded into cache/memory. Whenever user logs in, data is brought into cache. Initially it takes more time, but subsequent read/write operations will have improved performance. Middleware like CloudTran does not allow interactions with database directly. Therefore, data is fetched directly from the IMDG/database. CloudTran first writes data into cache and then writes back to database through backend process from time to time. After the user logs out, data is evicted from cache. This ensures that memory is free for the use by another application/data. The algorithm showing the steps involved in the proposed approach is given in Table 1.

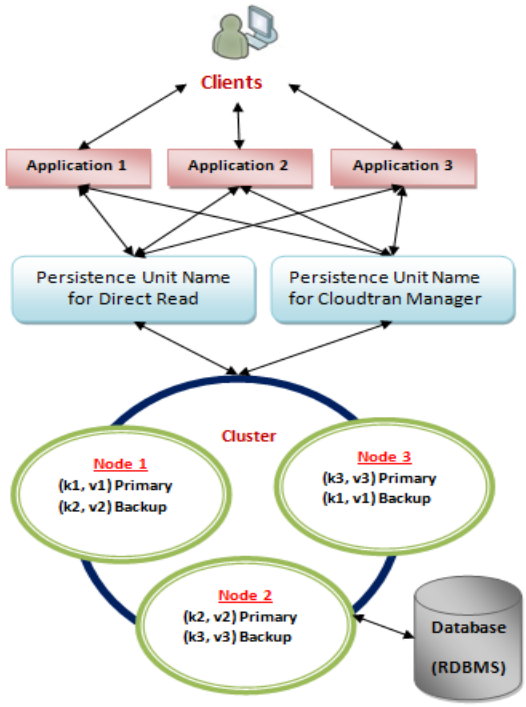

Fig 1: Proposed Usage of IMDG in Transactional Applications

4.2 Stale Data

To avoid stale data situation, There are four possibilities in any application architecture; single application single database, single application multiple databases, multiple applications single database and multiple applications multiple databases. In single application single database architecture, there is one application and data resides in one database server. In this kind of architecture, the problem occurs when data is updated directly into database bypassing application and IMDG. In single application multiple databases architecture, there is one application server having single application and data is stored across multiple databases. In this kind of architecture, one application access data from multiple sources of data. In multiple application single database architecture, more than one application access data stored in a single database. In multiple applications multiple database architecture, any number of application can access data stored in multiple databases. In Cloud, Database servers lie at remote location, therefore their access is restricted. The issue of multiple users accessing same data for update does not arise as only data relevant to the user is shown and loaded into memory.

Table 1. Algorithm for Reading and Writing Data

Phase	Steps
Reading Data	
Filtering Phase	Define criteria // like user name Retrieve data based on criteria // retrieves data directly from database Collection ← retrieved data
Loading Phase	For object = 1 to size of Collection Put data in cache using key and object End for
Writing Data	
Get object Write object to cache using CloudTran with unique key //CloudTran writes back to database automatically using write behind(Back) concept	

4.3 Data Caching in Cluster and ACID Guarantees

In cluster, copies of data are replicated or partitioned across multiple servers to ensure high availability and scalability. Providing ACID guarantees in a cluster becomes a challenge. To resolve this issue, middleware tools should be used which can provide ACID guarantees. Table 2 summarizes the issues and corresponding solutions to resolve them.

Table 2: Performance Issues and proposed solution

Performance Issue	Description	Proposed Solution
Initial Data	Whenever an application is started afresh, initial data is needed in the cache. To put application related initial data in cache, more time is required. But once data is in the cache, it can be accessed easily.	Only relevant data is read from database and stored in the cache whenever application is started first time. A policy is enforced for bringing only desired data in cache instead of whole set of data. It may be user dependent or application dependent.
Stale Data	Data stored in cache becomes stale when one application is accessing data from cache and other application or part of same application updates data directly on the database.	Direct updation of data in the database is avoided. In case, more than one application is updating data, optimistic lock can be used. Secondly, refresh the data in the cache time to time i.e. use cache invalidation. However, this process degrades the performance but it is must for maintaining consistency.
Data Caching in Cluster and ACID Guarantees	If data is stored at multiple locations in a cluster and it is updated at one location, same data available on other locations becomes out of date. So, transactional application cannot be deployed without ensuring ACID guarantees.	It is possible to avoid such situation by setting configuration files of IMDG tools, using JPA and by maintaining sessions. To ensure ACID guarantees, tools like CloudTran can be used on the top of any IMDG.

5. EXPERIMETATION WITH CASE STUDY OF EDUCATIONAL INSTITUTES

Education sector like any other sector uses transactional applications for the purpose of fast, efficient and reliable data processing. Student Information System, Fee Management, Library Management, Payroll and Personnel Information, Financial Accounting and Store Keeping applications are the commonly used transactional applications in educational institutes. An application of Student Return Registration System (SRRS) commonly used in educational institutes is taken as a pilot application to explore the role of IMDG in deploying transactional applications in Cloud environment. Most of the universities have affiliated colleges and departments where students are enrolled and their registration data is sent to the respective universities. Currently, affiliated colleges submit data of enrolled students for a particular session in the prescribed format as hard or soft copy to their respective universities for further processing. Few universities are also using browser based software to get this data from the affiliated colleges. A general discussion was held with lecturers and administrative staff of colleges of different universities. This interaction helped in understanding the flow of information and data exchanged between universities and colleges.

To verify the proposed approach, tools like CloudTran, Oracle Coherence and MySQL as backend database are used. An application is developed using Java and its features of JPA. The issue wise implementation of solution is described below.

5.1 Initial Data

Pilot application SRRS is meant for educational institutes like universities. A university has to interact with different users belonging to its affiliated colleges and departments. Therefore, logins are created for each college/department that can access data. University has its own login to access data of all colleges/departments. Since, registration process is done once a year, login ID and year are two main parameters that are required to filter data. Login IDs are linked to a particular college/department and college/department is further associated to a university. Since this application can be used for any university, data is filtered on the basis of three parameters- University, college/department and year. Only limited and desired data will be cached for college/department level users. When a university user logs in to have information on consolidated data of all of its affiliated colleges, two parameters are required; university and year. Views on database tables are useful for this type of scenario due to read only characteristics of query.

5.2 Stale Data

In case of university user, data is refreshed time to time. Cache Invalidation is better approach to avoid stale data. Data from cache is removed and is fetched from database again.

5.3 Data Caching in Cluster and ACID Guarantees

CloudTran runs over Oracle Coherence IMDG. JPA features are used to put data into cache/ memory across servers in a cluster. CloudTran middleware is used to ensure ACID guarantees while developing SRRS.

6. PERFORMANCE ANALYSIS OF PROPSOED APPROACH

The proposed approach to improve the performance using IMDG was applied on Student Registration Return System (SRRS). To verify the implementation of proposed approach, a small set of 1000 records of 40 characters variable length was prepared according to SRRS application. The database was created in MySQL. Records were read directly from the database as well as from the cache. The above steps are implemented in Java over CloudTran and Coherence with backend support of MySQL database. The observations were recorded for the time taken both for direct access of data from the database and the data reading from the cache. The time slightly varies as other processes were also running along with the application. The machine configuration includes Core i7 processor with 4GB RAM. However, time may vary from machine to machine. The time values taken to read data shows relative advantage of reading from cache when data is frequently required rather than having it from database again and again.

The developed program was executed on same set of data for ten times. During each execution, start and end time was noted. In the program, start time was displayed before fetching data. End time is displayed when program finishes task of fetching data and stores fetched data in memory. The time taken to read data from cache is almost 1/3rd than that of reading data directly from database. The average time for this experiment comes out to be 204.1 ms for cache read and 659.6 ms for database read. The time values taken by system are shown in Table 3. The results are graphically shown in Figure 2.

Table 3. Performance Analysis

Sr. No.	Direct from Database (Time in ms)	From Cache (Time in ms)
1.	641	236
2.	672	201
3.	710	162
4.	604	255
5.	799	267
6.	561	240
7.	671	153
8.	638	185
9.	692	151
10.	608	191
Average Time	**659.6**	**204.1**

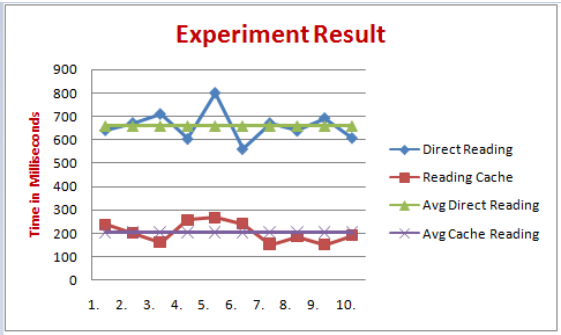

Fig 2. Comparison of Direct and Cached data accesses

From the above results, it is clear that when data is to be read frequently, it is better to load it into cache using IMDG like Oracle Coherence as done in SRRS application. For providing ACID compliance over write/update operations on data of educational institutes, middleware tools like CloudTran help in storing data into cache. Such tools also help in persisting data to database from IMDG using Write-behind approach while maintaining ACID rules.

7. CONCLUSION AND SCOPE FOR FUTURE WORK

Conventional databases are not able to deliver the performance level required by transactional applications deployed in Cloud. They are unable to scale with fluctuations in demand. IMDG has the ability to considerably improve the performance and scaling for transactional applications. It can be used to cut the costs of data scaling by reducing the dependence upon relational databases. Distributed nature of IMDG makes it suitable for transactional applications of any organization and helps them in achieving flexibility and high performance for their applications. Cloud Computing environment is used mainly for analytical applications. Therefore, it is explored to deploy transactional applications for educational institutes in Cloud while ensuring ACID guarantees with the help of IMDG, Oracle Coherence and middleware CloudTran. It is further concluded that future belongs to IMDG due to technology innovations, application requirements and their flexibility to work in diverse IT environments.

Current study uses simple data for experimentation. Same approach would be implemented using live data having more complex data sets, data relationships and applications for more generic interpretation.

8. REFERENCES

[1] Vasilios Andrikopoulos, Binz, Tobias, Leymann, Frank, Strauch Steve, "How to Adapt Applications for the Cloud Environment", In: Computing, Springer, vol. 95(6), 2013, Pages 493-535.

[2] N Ram Ganga Charan, S. Truiputi Roa, Dr. P.V.S. Sriniva, "Deploying Application in Cloud", International Journal of Advanced Computer Science and Application, vol 2, Issue 5, 2011, Pages 119-125.

[3] David Villegas, Ivan Rodero, Liana Fong, Norman Bobroff, Yanbin Liu, Manish Parashar, S. Masoud Sadjadi. The role of Gird Computing Technologies in Cloud Computing. Handbook of Cloud Computing , Springer Link, 2010, Pages 183-218.

[4] Razorfish, 2012, Using In-memory Data Gird to Bridge the Cloud, Gigaspaces

[5] Sushma R. Vhatkar, Sanchika A. Bajpai, "Throughput Genome Data Processing and Real - Time Analysis using Oracle Coherence In-Memory Technology", International Journal of Advanced Research in Computer Science and Software Engineering, vol 4, Issue 4, 2014, Pages 623-528.

[6] Brian Tierney, William Johnston, Jason Lee, 2000, A Cache based Data Intensive Distributed Architecture for Grid Applications, Lawrence Berkeley National Laboratory, Berkeley.

[7] Octavian Paul Rotaru, 2008, "Caching Patterns and Implementation", Leonardo Journal of Sciences, Issue 8,2008, Pages 61-76.

[8] John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Guru Parulkar, Mendel Rosenblum, Stephen M. Rumble, Eric Stratmann, Ryan Stutsman, "The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM", SIGOPS Operating Systems Review, vol. 43, Issue 4, 2009, Pages 92-105.

[9] Barkha Bahl, Vandana Sharma, Navin Rajpal ,"Boosting Geographic Information System's Performance using In-Memory Data Grid", BIJIT, BVICAM's International Journal of Information Technology, vol. 4, No. 2, 2012,Pages 468-471.

[10] Nick Kloski , Nitin Ramannavar, Satish Vanga, 2011, Oracle Optimized Solution for WebLogic Suite: An Optimal In-Memory Data Grid Architecture. An Oracle White Paper Version 1.1, Oracle Corporation.

[11] Luo Liqun, He Sijin , "A Memory Architecture Design for High-performance Cloud Computing", Advanced Materials Research, vol 532-533,2012, Pages 671-681.

[12] Dough Clarke, Andrei Badea, 2008, Developing Java Persistence API Application with the Netbeans IDE and EclipseLink, JavaOne Conference. Available at http://www.oracle.com/technetwork/ systems/ts-5400-159039.pdf, Last accessed October 1, 2014.

[13] Nilayam Kumar, Kamila, Renu Raghvan, Naveen Chalicheemala, "Object Caching Design for Improving Data Access Performance In enterprise Applications", International Journal of Computer Applications, vol 88, No. 13,2014, Pages 30-34.

[14] Qiong Luo, Sailesh Krishnamurthy, C. Mohand Hamid Piraheshd , Honguk Wooq, Bruce G. Lindsayd, Jeffrey F. Naughton, "Middle Tier Database Caching for e-business" in ACM SIGMOD International Conference on Management of Data, 2002, Pages 600-611.

[15] Understanding EclipseLink, 2.4. 2013, EclipseLink, available at http://www.eclipse.org/eclipselink/ documentation/2.4/eclipselink_otlcg.pdf, Last accessed July 10, 2014.

[16] Suvanam Sasidhar Babu, A. Chandra Sekhara Sarma, Yellepeddi Vijayalakshmi, N.V.Kalyankar, "Scalability of Multi Tier Transactions Towards Data Confidentiality For Cloud Applications", International Journal of Soft Computing and Engineering (IJSCE), vol. 2, Issue 4, 2012, Pages 247-250.

[17] Daniel J. Abadi, "Data Management in the Cloud: Limitations and Opportunities", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering: Vol. 32, No. 1, 2009, Pages 3-12.

[18] Arpita Mathur, Mridul Mathur, Palllavi Upadhyay,"Cloud Based Distributed Databases: The Future Ahead", International Journal on Computer Science and Engineering (IJCSE), vol. 3, No. 6, 2011, Pages 2471-2481.

[19] Data Layer Server for Web Applications, 2014, CloudTran.

[20] Joseph Ruzzi, 2013, Oracle Fusion Middleware Developing Applications with Oracle Coherence. Oracle Corporation.

[21] Java EE 6 Tutorial, 2013, Oracle Corporation. Mike Keith, Merrick Schincorial ,2009, Pro JPA2, Mastering the Java Persistence API. Apress.